RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPŔIEUR ET DE LA RECHERCHE SCIENTIQUE
UNIVERSITÉ DJILALI LIABÈS DE SIDI BEL-ABBÈS

FACULTÉ DE TECHNOLOGIE
DÉPARTEMENT D'INFORMATIQUE

Une approche instrumentée pour la mise en œuvre des Systèmes d'Informations Pervasifs

THÈSE

Pour l'obtention du

DOCTORAT EN SCIENCES DE L'UNIVERSITÉ DJILALI LIABÈS DE SIDI BEL-ABBÈS

Spécialité Informatique

par :

Mohammed Fethi KHALFI

Composition du jury :

Presidént :

Mimoun MALKI	Professeur, Universitè de Sidi Bel Abbès.

Examinateurs :

Ahmed LEHIRECHE	Professeur, Universitè de Sidi Bel Abbès.
Bouabdellah KECHAR	Maître de confèrences A, Universitè d'Oran .
Amine ABDELMALEK	Maître de confèrences A, Universitè de Saida.
Djelloul BOUCHIHA	Maître de confèrences A, Centre.Universitaire de Naâma.

Directeur :

Sidi Mohammed BENSLIMANE	Maître de confèrences A, Universitè de Sidi Bel Abbès.

Remerciement

La thèse est une aventure extraordinaire, une belle expérience, difficile par moment et agréable par d'autres. J'ai vécu mon doctorat comme une véritable aventure, un enrichissement personnel et professionnel considérable. C'est l'aboutissement d'une étape importante dans ma vie et le commencement d'une autre que j'espère aussi attractive. Un énorme Merci à toutes les personnes qui ont contribué pour que cette expérience soit si exceptionnelle.

Je tiens à remercier en premier lieu monsieur, Sidi Mohammed BENSLIMANE, Professeur à l'Université Djillali LIABES et directeur de l'école nationale d'informatique de Sidi Bel Abbes pour m'avoir accordée sa confiance en acceptant de diriger mes recherches. Je le remercie infiniment de m'avoir fait bénéficier tout au long de ce travail de sa grande compétence, de sa rigueur intellectuelle et de ses précieux conseils. Sa revendication constante du travail sérieux m'a aidée à progresser. Que le fruit de ces longues années de travail soit à la hauteur de ce qu'il a semé en moi.

J'exprime également toute ma gratitude à mon père Ali KHALFI, Professeur et Recteur de l'université de Sidi Bel Abbes qui m'a épaulé tout au long de ma carrière universitaire et prodigué de précieux conseils qui m'ont permis d'aller constamment de l'avant. Il m'a encouragé par ses orientations sans cesser d'être une grande source de motivation et de persévérance. Son exigence m'a permis de progresser dans l'élaboration de ma recherche. Qu'il trouve dans ce travail l'expression de ma profonde et sincère reconnaissance.

Je souhaiterais adresser mes remerciements les plus sincères à monsieur Mimoun MALKI Professeur à l'Université de Sidi Bel Abbes, pour m'avoir fait l'honneur d'être président de jury. Je voulais également remercier les examinateurs de ce travail, monsieur Ahmed LEHIRECHE, Professeur à l'Université de Sidi Bel Abbes, monsieur Bouabdellah KECHAR, docteur à l'Université d'Oran,monsieur Amine ABDELMALEK, docteur à l'Université de Saida et monsieur Djelloul BOUCHIHA docteur au centre universitaire de Naama pour avoir accepté de faire partie de mon jury de thèse et d'évaluer mon travail. Je les remercie pour leurs applications et leurs précieux commentaires et recommandations.

Cet aboutissement n'aurait jamais pu se faire sans l'encouragement, le soutien et l'amour de tous les membres de ma famille et de mes amis. A ma cher Maman, à mes frères : Merci d'avoir cru en moi, Merci pour votre patience et pour l'affection que vous m'avez manifestée durant ces années.

Merci du fond du coeur pour votre soutien et pour vos précieux conseils, Une dédicace à ma femme et une toute particulière à ma fille Meriem et mon fils Ibrahim..
Pour finir, je souhaiterais dédier ce travail à tous ceux que j'aime et qui ne font plus partie de ce monde <Rabbi Yerhamhoum>. Qu'ils y trouvent ici l'expression de ma profonde affection.

Resumé

Grace à la démocratisation des technologies mobiles et le développement des réseaux sans fils, l'homme est entouré d'un réseau d'objets doté des caractéristiques d'un ordinateur : téléphones portables aux fonctions multiples, assistants personnels, ordinateurs embarqués de voiture, smartphones et tablettes. Les Systèmes d'Information (SI) n'échappent pas à cette évolution, donnant ainsi naissance à de nouveaux systèmes d'information dits pervasifs ou ubiquitaires (SIP). A la différence des systèmes d'informations classiques, ils apparaissent aujourd'hui dans un contexte de miniaturisation du matériel, d'ubiquité de l'informatique et d'essor des réseaux de communication sans fil. L'information peut être partagé entre les utilisateurs et consulté depuis n'importe quel terminal, à n'importe quel endroit et à n'importe quel moment. Un des défis actuel est de développer et d'utiliser ces nouvelles technologies non seulement pour fournir des services centrés utilisateur et contextualisés mais également pour transformer un SI donné en un SIP. L'objectif de la thèse est de proposer une méthode instrumentée pour introduire la pervasivité dans un système d'information, pour cela :

En premier, à partir de l'analyse de la littérature du domaine, nous abordons la convergence des systèmes d'informations classiques vers les SIP. Nous dressons une liste de défis et challenges de l'informatique ambiante pour définir les concepts élémentaires d'un modèle architectural générique qui forme les éléments de base d'un environnement pervasif.

Seconde, la notion de contexte représente les informations capables d'être observées dans cet environnement. Ainsi, les systèmes d'informations intégrés dans un tel environnement pervasif se doivent être sensibles au contexte afin de s'adapter à la dynamique de l'environnement. De plus, afin de représenter les informations contextuelles capturées et de s'adapter en conséquence, ces systèmes doivent définir au préalable un modèle de contexte. C'est l'un des éléments fondamentaux pour la mise en place de SIP. Nous présentons un méta modèle du contexte basé sur une ontologie pour les environnements ubiquitaires qui sera utilisé pour offrir des services de personnalisation sensibles au contexte.

A la fin, nous présentons une nouvelle vision proactive et contextuelle des SIP. Une approche web services pour dispositifs déployés dans une infrastructure UPnP. Elle se base sur l'orientation service, la sensibilité au contexte et sur une approche proactive afin de proposer des services pertinents et implicitement aux utilisateurs et dispositifs.

Mots clés : *Systèmes d'Informations Pervasifs, Systèmes Ambiants, Architecture des systèmes d'informations pervasifs, Informatique Ubiquitaire, Ontologies, Contexte, Sensibilité au Contexte, Modélisation du contexte, Web Services, Web Services pour dispositifs, Service Proactive, SOA, UPnP.*

Abstract

The recent technological overhangs have focused on the democratization of wireless networks and the miniaturization of communication devices. In this context, Ubiquitous Computing is a recent paradigm whose objective is to allow users to access data, and make information available anywhere and at any time. In other terms, Pervasive Information Systems (PIS) constitute an emerging class of Information Systems where Information Technology is gradually embedded in the physical environment, capable of accommodating user needs and wants when desired. PIS differ from Desktop Information Systems (DIS) in that they encompass a complex, dynamic environment composed of multiple artefacts instead of Personal Computers only, capable of perceiving contextual information instead of simple user input, and supporting mobility instead of stationary services.

First, as an initial step, we present PIS novel characteristics compared to traditional desktop information systems ; we explore this domain by offering a list of challenges and concepts of ubiquitous computing that form the core elements of a pervasive environment. As a result of this work, a generic architecture for intelligent environment has been created. Based on various and related works concerning models and designs. This framework can be used to design any PIS instance.

Second, Context awareness is the core feature of pervasive computing. High-level context awareness can be enhanced by situation awareness that represents the ability to detect and reason about the real-life situations. In order to deal with the problem in context-aware modeling in pervasive computing environments, we present a comprehensive and integrated approach for context modeling. We first propose a Meta model context based on ontology for Pervasive Computing aiming to extend its capabilities by adding new environmental aspects. The advantage is that it can provide a flexible modeling mechanism for multiple applications of context-aware pervasive computing.

Finally, Pervasive environments are characterized by rich and dynamic context, where users need to be continuously informed about services relevant to their current context. We will present an enhancement of ubiquitous computing discovery mechanisms adding context handling capabilities to Web Services for Devices in Pervasive Computing using UPnP as an infrastructure to address these implicit requests. Our method uses location aware, UPnP infrastructure, web service for devices and the notion of proactivity in pervasive computing to continuously present the most relevant services to the user or device in response to changes of context, services or user preferences.

Keywords : *Pervasive Information Systems, Design Pervasive Systems, Ambient Intelligence, ubiquitous computing, smart space, Web Services, Web Services For Devices, Proactive Service, SOA, UPnP, Context Awareness, Pervasive computing, Context Modeling, Ontology.*

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Le travail réalisé dans le cadre de cette thèse se situe dans le domaine des Systèmes d'Information Pervasifs (Ubiquitaires), en effet, ces systèmes sont au carrefour de plusieurs domaines technologiques (multimédia, réseau, systèmes distribués et embarqués) et préoccupations sociales (confidentialité, fiabilité, sécurité). Déja nos environnements et nos vies sont remplis d'une multitude d'appareils informatiques qui se contentent de jouer chacun leur role indépendamment les uns des autres et inconscients du monde qui les entoure. L'informatique pervasive vise à les orchestrer pour constituer un tout cohérent, un réseau d'appareils interconnectés et communiquants, jouant désormais leur role dans une piéce dont ils sont tous acteurs. Le but final de cette comédie est de tapisser l'environnement de l'utilisateur d'une couche transparente d'intelligence, capable de le comprendre et de lui être utile partout et tout le temps, sans le perturber. Les dispositifs unitaires deviennent alors intègrès à l'environnement et à notre vie quotidienne, interagissant avec l'utilisateur d'une manière naturelle, presque en arrière-plan de son attention.

Ce document présente une approche instrumentée composite de divers orientations permettant de concevoir et de mettre en œuvre la nouvelle génération des systèmes d'information, qu'on appelle **Systèmes d'Information Pervasifs (SIP)**.

1 La vision d'un système d'information pervasif

1.1 Informatique Pervasive

Depuis plusieurs années, nous assistons a une évolution considérable des technologies et des habitudes d'utilisation de l'informatique. Nous sommes témoins de l'évolution de l'usage de ces technologies afin d'accéder aux différents systèmes qui prennent place dans notre quotidien. En effet, la démocratisation des dispositifs utilisés dans notre vie quotidienne (Smartphones, tablettes, etc.), couplée aux avancées technologiques mobiles (3G, géolocalisation, Bluetooth, etc.) ont bouleversé la manière dont on utilise ces systèmes. L'informatique devient alors disponible partout : c'est l'informatique pervasive. Les smartphones, par exemple, sont considérés comme la plateforme informatique de l'avenir, où l'informatique peut être fortement décentralisée et répartie sur les différents terminaux utilisés par les utilisateurs pour fournir des services hautement pertinents [Ker11].

Ces avancées technologiques, qui nous donnent le pouvoir d'interagir avec le monde d'une manière naturelle, ont contribué à l'amélioration de nos capacités quotidiennes. L'informatique s'est intégrée à l'environnement d'une façon

invisible. Par exemple, l'usage du GPS dans différentes situations, [Sch10] utilise trois récepteurs GPS différents : le premier dans son téléphone, fournissant aux réseaux sociaux sa localisation, le second intégré dans le système de navigation de la voiture et le dernier dans l'appareil photo, afin de géo-localiser les photos prises.

Les systémes pervasifs désignent désormais l'informatique nouvelle ou l'environnement est naturellement doté de capacités de traitement et de communication, intégrées de façon à devenir invisibles et omniprésents. La finalité de tels systémes tournés vers l'utilisateur est de fournir un environnement personnalisé qui facilite l'accés aux services et améliore leur utilisation de façon transparente.

L'informatique ubiquitaire (ou Pervasive) est une réalité et non pas un avenir proche. Elle fait partie intégrante de notre quotidien, notamment à travers les nouvelles technologies [BD07]. Celles-ci sont devenues presque invisibles, à tel point qu'il nous est désormais impossible d'imaginer notre vie personnelle et professionnelle sans elles. L'informatique ubiquitaire a pris une forme différente de celle attendue par Weiser [Wei91], dans laquelle les dispositifs mobiles représentent l'élément central de notre vie quotidienne. L'auteur soutien que nous sommes continuellement en train d'utiliser des ressources informatiques dans notre vie courante sans forcément les percevoir en tant qu'ordinateurs. L'informatique ubiquitaire est ainsi une réalité sous la forme d'un environnement densément peuplé de ressources informatiques et de communication.

1.2 Impact de l'informatique pervasive sur les systemes d'information

La dernière décennie a été marquée par le changement dans la manière dont nous travaillons et dans la manière dont nous nous appuyons sur les technologies. Nous passons d'un modèle statique, dans lequel les travailleurs interagissent dans des circonstances bien définies (ordinateur de bureau), à un modèle dynamique, dans lequel ils se caractérisent par leur mobilité, permise par l'évolution des réseaux sans fil et des dispositifs mobiles.

Ainsi, les Systèmes d'Information (SI) sont confrontés à un environnement pour lequel ils n'ont pas été prévus. En effet, l'arrivée de l'informatique ubiquitaire, au sein des entreprises, a directement impacté les SI. La mobilité qu'apportent ces nouvelles technologies a étendu les SI bien au-delà des frontières physiques des entreprises. Ceci revient à dire que l'évolution de ces technologies mobiles et pervasives a ouvert de nouvelles perspectives et a changé le mode d'accès à ces systèmes. Nous assistons au passage graduel d'un paradigme entièrement fondé sur les desktops à un paradigme mixte, intégrant des dispositifs multiples et très hétérogènes : desktops, dispositifs mobiles et ressources intégrées à l'environnement physique [KG06].

Depuis leur apparition, les systèmes d'information n'ont cessé d'évoluer et de progresser dans la perspective d'améliorer la productivité et l'efficacité au sein de l'entreprise, comme l'illustre la Figure 1. Au début, un système d'information a été défini comme un ensemble organisé de ressources (individus, matériel, logiciel, progiciel, bases de données, procédures) qui permettent d'acquérir, de traiter, de stocker, et de communiquer l'information sous différentes formes au sein d'une organisation [Rei04].

Avec le développement des nouvelles technologies, les SI sont confrontés à de nouveaux modes d'interaction . L'évolution devient ainsi inévitable. Ces systèmes doivent s'adapter aux nouvelles technologies et aux nouveaux modes d'accès mis à disposition de leurs utilisateurs. Nous assistons donc aujourd'hui à l'émergence d'une nouvelle génération de systèmes d'information : **les Systèmes d'Information Pervasifs (SIP)**. Nous assistons aujourd'hui au passage progressif des technologies de l'information (IT) à l'arrière-plan. En d'autres termes, les SI sont là aujourd'hui pour surveiller les activités des utilisateurs, pour assembler et traiter les informations et pour intervenir lorsque cela est nécessaire [KG06].

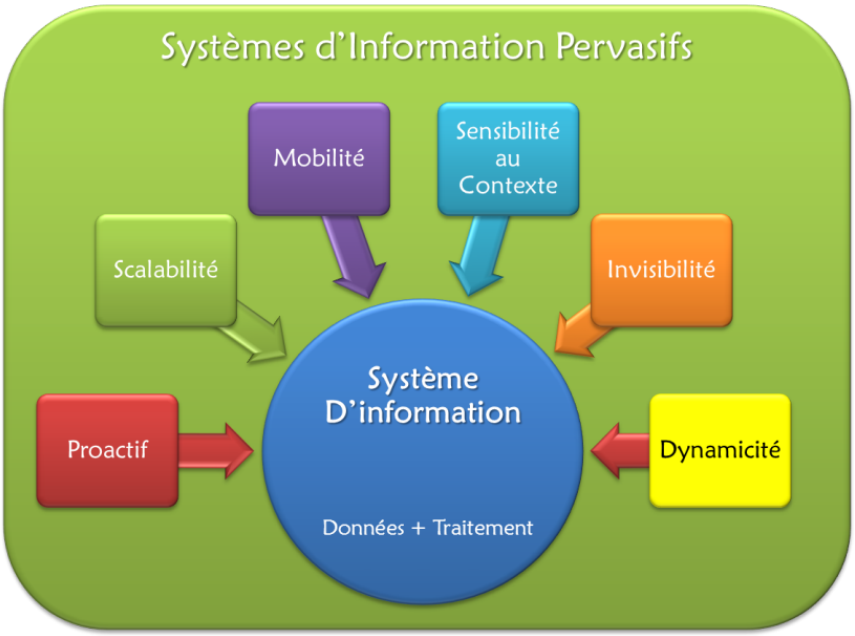

Figure 1 – L'émergence des Systèmes d'Information Pervasifs.

1.3 Les systèmes d'information pervasifs (SIP)

La notion de système d'information pervasif (SIP) représente une nouvelle classe des SI et assure un avenir prometteur. Elle apporte de nouvelles opportunités, notamment par la prise en compte de l'environnement et la possibilité d'offrir des services innovants. A l'inverse des SI traditionnels, les SIP s'intègrent progressivement à l'environnement physique [KG06]. Contrairement aux SI traditionnels, les SIP s'ouvrent à d'autres dispositifs plus évolués, offrant à l'utilisateur des espaces plus étendus d'interaction avec le SI : des dispositifs mobiles transportables par l'utilisateur, dispositifs directement intégrés à l'environnement. Ainsi, un SIP peut intégrer des éléments d'interactions mobiles avec des dispositifs ou des objets de l'environnement physique d'une manière naturelle et discrète.

La vision de Marc Weiser est devenue aujourd'hui possible grâce à la combinaison de trois facteurs principaux : la miniaturisation et la puissance des composants électroniques, la chute des coûts de production et l'omniprésence des technologies réseaux sans fil [Wei91]. En effet, ces dernières années ont vu la miniaturisation des dispositifs informatiques et électroniques dotés de capacités de capture, de traitement et de communication, pendant que leur puissance de calcul continuait d'augmenter. Beaucoup de progrès ont également été réalisés afin de réduire leur consommation d'énergie. Cette situation, associée à la chute des coûts de production, a favorisé la démocratisation de ces technologies et leur intégration dans de nombreux objets du quotidien (les téléphones, appareils photos et caméras vidéo, les lecteurs et instruments de musique, les contrôleurs enfouis dans les avions, les voitures ou encore l'électroménager deviennent des ordinateurs habillés autrement). En outre, le développement des technologies réseaux sans fil (e.g., Bluetooth, Wi-Fi, 3G+) a permis à ces nouveaux types d'objets communicants d'interagir spontanément avec l'utilisateur ou avec d'autres objets, aussi bien localement qu'à distance, et de favoriser leur mobilité.

Un système pervasif permet d'automatiser certaines tâches quotidiennes grâce aux différents objets communicants disponibles [SM03]. La figure 2 représente le fonctionnement d'un tel système ubiquitaire. Le système collecte tout d'abord des informations de l'environnement physique (e.g. température ambiante, lumière du soleil, bande passante, présence d'un utilisateur) à l'aide d'objets communicants capables de les capturer. De tels objets communicants sont

appelés des capteurs. Ces derniers peuvent être aussi bien physiques que logiciels. Les informations collectées sont ensuite interprétées, filtrées et agrégées par diverses applications, afin de les enrichir de données contextuelles. À partir de ces dernières, certaines applications peuvent décider des actions à entreprendre (e.g. allumer une lumière, déclencher une alarme, modifier un statut, afficher une information) par les objets communicants capables d'agir sur l'environnement physique. De tels objets communicants sont appelés des actionneurs.

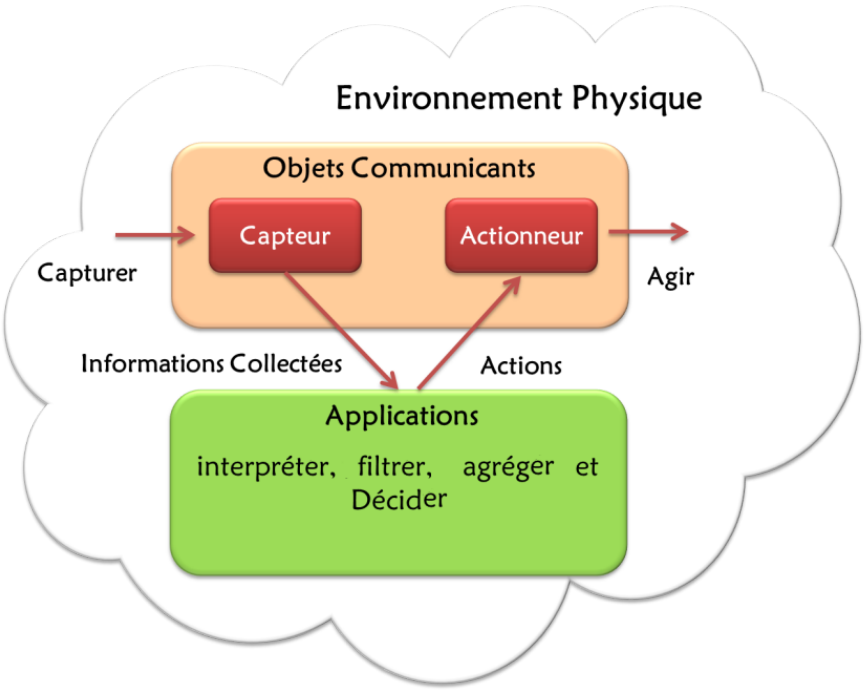

Figure 2 – Fonctionnement d'un système d'informatique Ubiquitaire.

Les exemples de systèmes d'informatique ubiquitaire sont nombreux. Nous pouvons citer la détection précoce par réseaux de capteurs des accidents écologiques (e.g. incendies), la gestion intégrée des bâtiments (e.g. consommation d'énergie, sécurité) ou encore la surveillance des personnes âgées peu autonomes. Considérons l'exemple d'un système de sécurité pour se protéger contre les intrusions. Ce système est responsable de détecter et de signaler toutes les tentatives d'intrusion d'un individu dans un bâtiment. Pour cela, il peut nécessiter différents types d'objets communicants, tels que des détecteurs de mouvements pour collecter les coordonnées d'une intrusion, un agenda pour connaître les plages horaires de surveillance du bâtiment, diverses applications pour analyser les données collectées et déterminer si une intrusion a bien lieu afin d'y réagir, des alarmes sonores pour alerter les alentours, des caméras vidéo pour filmer l'intrusion, une plate-forme de téléphonie pour envoyer la vidéo de l'intrusion sur le téléphone portable du propriétaire, etc. Cet exemple donne une bonne idée de la taille et de la complexité qu'un système d'informatique ubiquitaire peut avoir, coordonnant une multitude d'objets communicants et requérant une expertise étendue dans de nombreux domaines, comme la programmation distribuée ou encore les télécommunications.

1.4 La sensibilité au contexte dans les systèmes d'information pervasifs

Les SIP s'inscrivent dans un environnement particulièrement dynamique, composé d'une multitude d'artefacts capables de percevoir le contexte de l'utilisateur et de gérer sa mobilité [KG06]. D'autre part, ces SIP offrent, par

cette intégration à un environnement hétérogène et une interaction continue avec le SI là où on est et quand on le souhaite. Contrairement aux SI traditionnels, dont l'intelligence réside dans l'ordinateur desktop, les SIP doivent faire résider cette intelligence au-delà de l'ordinateur en l'intégrant dans le monde physique. De plus, les SIP doivent être proactifs, réagissant aux stimuli de l'environnement, à l'encontre des SI traditionnels, dans lesquels une réponse du système est forcément précédée d'une action de l'utilisateur. La prise en compte du contexte d'utilisation dans les applications est un domaine de recherche d'actualité connu sous le nom de **sensibilité au contexte** ou **context-Awareness** en anglais. Une application sensible au contexte doit percevoir la situation de l'utilisateur dans son environnement et adapter par conséquent son comportement à la situation en question.

2 Contexte

Depuis toujours, l'informatique ne cesse d'évoluer et nous sommes passés du stade de l'ordinateur central associé à plusieurs individus au stade de la multiplication des ordinateurs pour une même personne. Les ordinateurs sont devenus plus petits et plus performants. Ils sont aussi embarqués dans les objets de la vie quotidienne, il s'agit de l'informatique ambiante (intégrée dans les objets ambiants enrichis de capacités de traitement de l'information). Grâce aux technologies sans fils, les objets sont interconnectés et communiquent pour tout échange d'informations. Nous parlons dans ce cas d'informatique ubiquitaire où l'information est accessible de n'importe où et n'importe quand.

Mark Weiser [Wei91] a déclaré en 1991 dans son livre The computer for the 21st century, "les technologies les plus réussies sont celles qui disparaissent. En effet, elles s'associent à la vie de tous les jours jusqu'à ce qu'il devienne difficile de les discerner". Depuis, un développement significatif a marqué l'informatique grâce aux avancées perçues dans les techniques de communication, les communications sans fils et les capteurs. Ainsi, la vision de Weiser a prouvé son exactitude surtout avec l'apparition de l'informatique ubiquitaire ou pervasive. L'objectif des systèmes ubiquitaires est d'accéder à tout moment, de tout endroit et avec tout média à toute information et à tout service. Ainsi les objectifs des systèmes informatiques récents ne sont plus restreints à l'exécution de tâches commandées par l'utilisateur mais à faire communiquer plusieurs systèmes mobiles ou fixes pour fournir des services personnalisés à un seul usager.

La vision de Mark Weiser est que la multiplication des systèmes va changer radicalement notre façon de les utiliser [Wei91]. Les applications ne seront plus associées à une machine physique et surtout un ècran associèe. Elles vont pouvoir migrer et nous suivre au grè de nos dèplacements. Les interactions vont etre plus naturelles et l'ordinateur va se fondre dans notre environnement et disparaitre. L'informatique ubiquitaire peut etre vue comme l'opposé de la rèalitèe virtuelle. La rèalité virtuelle met une personne à l'intérieur d'un monde créé par l'ordinateur. L'informatique ubiquitaire a pour but de permettre à l'ordinateur de vivre dans le monde des hommes et de s'y intégrer au point de disparaitre.

Les SI d'aujourd'hui affrontent le défi de supporter la mobilité pour permettre de gérer les informations depuis ou vers des dispositifs mobiles. Pour permettre l'amélioration de la qualité des services, l'usage des technologies ubiquitaires permet la prise en compte du contexte. Il s'agit en fait d'une qualité à intégrer aux SI pour augmenter leur niveau d'intelligence et d'autonomie. La prise en compte du contexte est la perception de l'environnement pour interagir plus naturellement avec l'utilisateur. Pour supporter cette fonctionnalité, des données en relation avec le contexte sont utilisées telles que les données des capteurs de l'environnement physique, des matériels auto-descriptifs, la description des personnes ou encore les métadonnées sur les applications. Les exigences issues de la

mobilité représentent de nouveaux défis qui doivent être considérés et traités par le SI. La détection des évènements survenant dans l'environnement d'exécution est primordiale pour les applications mobiles pour qu'elles soient sensibles au contexte. Pour être réactifs au contexte, ces SI doivent réagir à ces évènements de façon efficace et en temps réel.

Un SI ubiquitaire doit assumer certaines caractéristiques telles que la Scalabilité, l'invisibilité et la pro-activité. La Scalabilité désigne le passage à l'échelle et la capacité de gérer un nombre croissant d'utilisateurs, d'applications et de matériels. L'invisibilité nécessite un minimum d'intervention humaine. La pro-activité est la capacité d'analyser les situations et les problèmes pour suggérer et proposer à l'utilisateur des solutions en réponse à son contexte actuel ou prévu. Les caractéristiques de ces systèmes ubiquitaires et leurs capacités de traitement et de communication élargissent leur spectre d'application et justifient le nombre d'applications qui ne cesse de croître.

Les utilisateurs et leurs terminaux portables sont mobiles, et peuvent se trouver n'importe où. La localisation est devenue, de nos jours, l'un des éléments les plus importants de l'informatique, et aussi l'un des aspects le plus étudié. Un très grand nombre d'applications dépend de cette fonctionnalité pour fournir des services pertinents aux utilisateurs. Parfois on a juste besoin de savoir quand on est très proche d'une certaine entité. Comme celles de visites guidées de musées, devant les oeuvres se trouve une puce RFID destinée à initier une explication audio au moment où le visiteur s'approche.

3 Problématiques

Dans nos travaux, nous nous intéressons à la conception et la mise en œuvre d'un SIP sensible au contexte. Ce système est capable de développer et d'utiliser des nouvelles technologies pour fournir des services pertinents centrés utilisateur. Les SIP constituent une classe émergente des systèmes d'information classique où La technologie est intégrée progressivement dans l'environnement physique.

Problématique 1 :Migrer du paradigme systèmes d'information classique (Desktop) vers des systèmes d'information pervasifs (SIP) :

Le SIP emploi une multitude de types de dispositifs hétérogènes qui diffèrent en termes de taille, forme (diversifié, ergonomique et style) et fonctionnalité (simples téléphones mobiles, portables ordinateurs portables, assistants numériques personnels, des capteurs). Ce SIP utilise des dispositifs mobiles, opposé aux environnements de bureau où ils sont stationnaires.

En outre, le SIP revoit la façon dont nous percevons la conception des systèmes d'informations. Dans les environnements classiques, les profils d'utilisateur sont connus à l'avance. Les SIP intègrent de plus en plus le monde physique. Avant le logiciel était conçu pour exploiter un matériel et aujourd'hui il est un moyen pour effectuer une tâche prédéfinie. Cette tâche peut dépendre d'un emplacement géographique ou peut être déclenchée par un événement comme une demande d'utilisateur. L'utilisateur interagit avec de nombreux objets informatiques au lieu d'un seul ordinateur (environnement intelligent) sans une formation préalable. Enfin, le SIP introduit la notion de la sensibilité aux contextes suite aux objets intelligents à capacité de recueillir, traiter et gérer les informations de l'environnement. Opposé au système classique, où l'action de l'utilisateur précède la réponse du système, les SIP sont des systèmes proactifs capable d'anticiper les besoins de l'utilisateur dans le monde physique.

L'informatique pervasive exige, la mise en place de nouveaux outils et de nouvelles mesures (architectures, cadres de travail, intergiciels) qui ne faisaient pas partie de l'informatique traditionnelle. La plupart des outils actuels ne

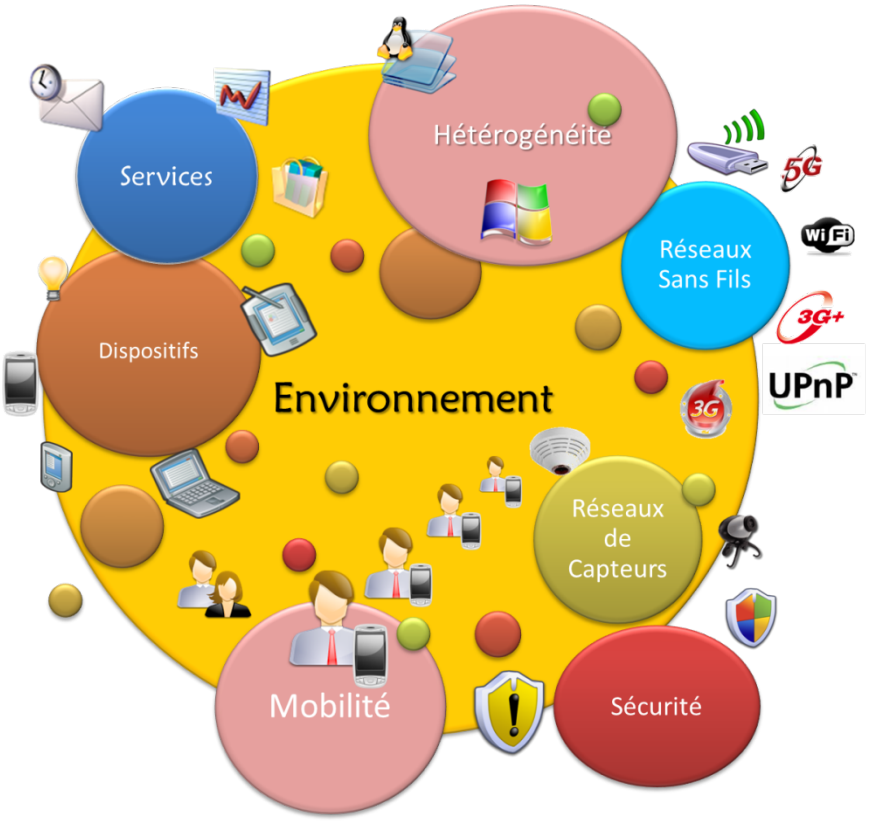

Figure 3 – Défis et challenges d'un environnement pervasif.

tiennent pas compte des particularités des SIP telles que la mobilité et la sensibilité au contexte. Ces outils seront d'une grande importance pour appuyer la conception des systèmes pervasifs sensibles au contexte, Figure 3.

Problématique 2 : Concevoir un Model Architecturale d'un environnement pervasif :

Les SIP constituent une nouvelle génération des SI qui est difficile à conceptualiser, avec peu de formalismes disponibles. Partant de ce fait, il est nécessaire de mettre en place un cadre et une architecture plus formelle permettant d'aider les concepteurs à mieux comprendre les SIP et surtout à mieux les maîtriser, tout en assurant la transparence nécessaire à ces systèmes. En étudiant la vision actuelle des systèmes ubiquitaires à travers les différents travaux de la littérature et les systèmes existants récemment conçus, nous remarquons que les architectures actuelles offrent très peu d'abstraction et sont spécifiques à une application ou à un domaine particulier. Ils ne sont pas suffisamment génériques pour être réutilisables dans d'autres domaines. Cette problématique est illustrée dans la Figure 4.

Les SIP se caractériseraient par l'interaction continue rendue possible par des dispositifs mobiles intégrés à l'environnement physique, ce qui ouvre la possibilité d'offrir aux utilisateurs de nouveaux services innovants. Toutefois, afin de concevoir de tels SI, il est nécessaire de bien comprendre les caractéristiques et les exigences auxquelles ces nouveaux systèmes seraient soumis. Tels que l'hétérogénéité des ressources, des infrastructures et des terminaux. Le dynamisme de l'environnement varie en fonction de l'utilisateur (ses actions, sa mobilités, etc.) et de ses éléments,

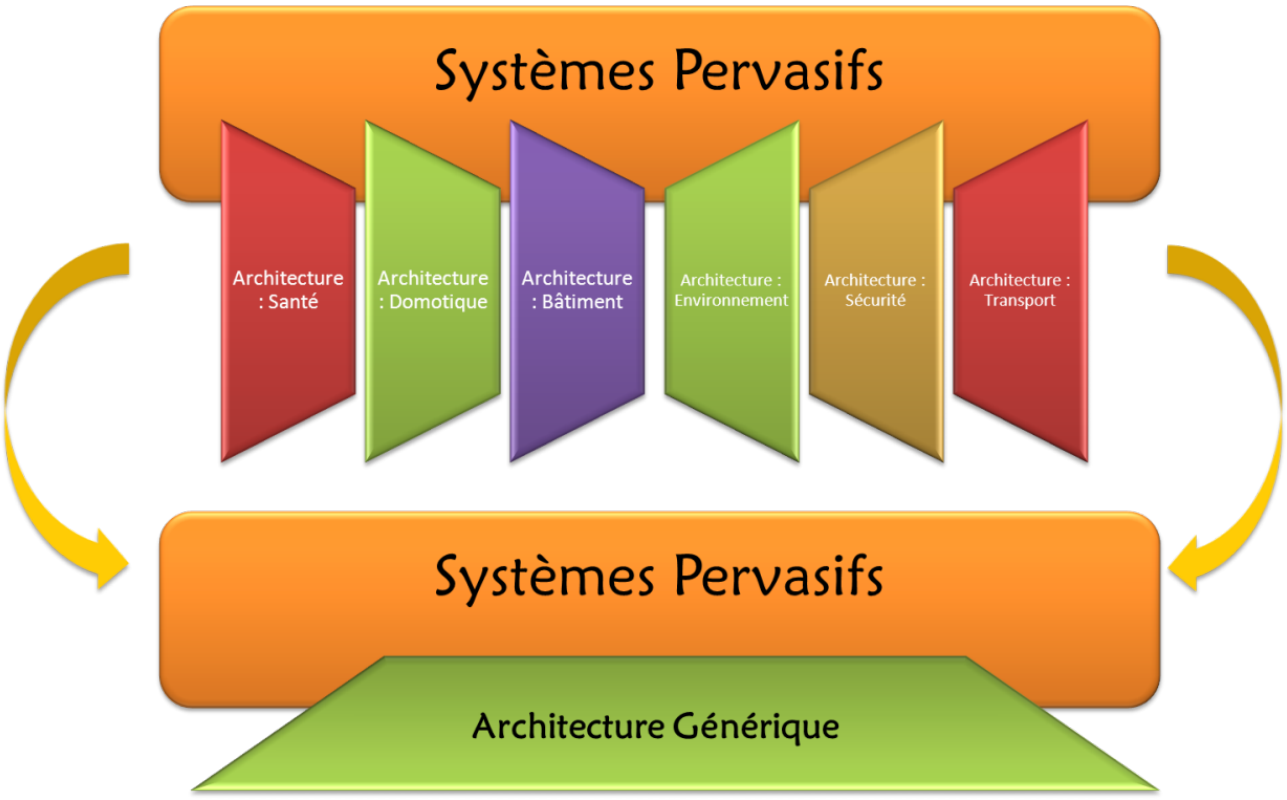

Figure 4 – Convergence vers un système d'information Ubiquitaire Générique.

Figure 3. Ainsi, nous pouvons résumer notre problématique à un problème de conception et de réalisation d'un SIP qui répond à tous les besoins de transparence et d'adaptation à l'environnement. En effet, avec le manque de modèle permettant de prendre en compte tous ces besoins, le concepteur du SIP se trouve démuni face à une nouvelle génération de SI qui jusqu'à présent n'a pas été véritablement mise en place avec des formalismes appropriés dans l'objectif d'aider et d'orienter les concepteurs.

Problématique 3 :Introduire un modèle de contexte au sein de notre système :

La modélisation du contexte est une étape essentielle pour le développement des applications pervasives sensibles aux contextes. La plupart des approches proposées dans la littérature sont ou bien spécifiques à un domaine particulier ou bien il leurs manque le formalisme nécessaire pour la modélisation. Ces approches ne sont pas suffisamment génériques pour être réutilisables dans d'autres applications ou autres domaines et ils ont un niveau de reconfiguration limité.

Afin d'assurer la transparence nécessaire, les SIP doivent être sensibles au contexte, permettant la prise en compte de l'environnement pervasif. Dans ce cadre, la notion de contexte représente l'ensemble des caractéristiques de l'environnement physique ou virtuel qui affectent le comportement d'une application et dont la représentation et l'acquisition sont essentielles à l'adaptation des informations et des services. La sensibilité au contexte représente la capacité d'un système à percevoir les informations contextuelles relatives à l'utilisateur, au système lui-même et

à l'environnement afin de pouvoir adapter ses fonctionnalités de manière dynamique et proactive, réagissant aux stimuli de l'environnement. Ainsi, il est nécessaire de représenter le fonctionnement global du système à travers une modélisation de haut niveau.

4 Aperçu de la proposition

L'informatique ubiquitaire est constituée d'applications logicielles s'exécutant dans un espace peuplé de dispositifs communicants de la vie de tous les jours. Ces dispositifs sont informatisés et sont capables de construire un contexte pour l'application en fournissant à leur environnement des informations sur leur état et également en agissant sur leur environnement. La détection, l'analyse et l'adaptation à des situations contribuant à cette perception sont des tâches inconscientes que nous avons l'habitude d'effectuer tout au long de la journée. La contribution de cette thèse est double. Dans un premier temps nous présentons une analyse complète des caractéristiques et des challanges de l'informatique pervasive. Ensuite nous proposons une mise en oeuvre qui se base sur une la sensibilité au contexte, un modéle service pour dispositifs et une orientation proactive afin de résoudre les problèmes de transparence et d'adaptation à l'environnement.

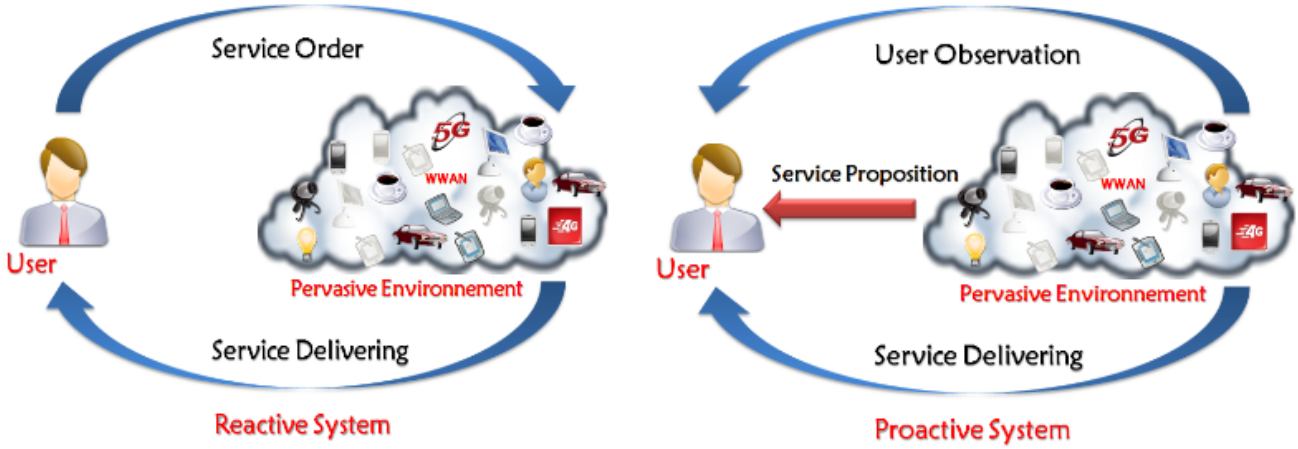

Figure 5 – Systèmes réactifs et proactifs au sein d'un système d'informatique ubiquitaire.

Notre intérêt porte en particulier sur la proposition d'un nouveau modèle architectural générique pour appuyer la conception des systèmes pervasifs. Nous proposons une solution pour aider à concevoir un SIP en présentant un cadre conceptuel décrivant et formalisant l'ensemble de ses éléments. Dans ce cadre, en se basant sur la notion de contexte et de service et en exploitant la relation qui les lie, nous proposons une nouvelle vision proactive d'un SIP. Notre vision est déclinée sur les dimensions suivantes :

1. Dimension conceptuelle : Une infrastructure permet l'échange des informations de contexte, suffisamment générale pour être utilisée par différentes applications pervasives, et suffisamment spécifique pour couvrir les informations fournies par les dispositifs communicants.
2. Dimension fonctionnelle : Grâce aux mécanismes de découverte de services proactifs, la nature hautement dynamique des environnements pervasifs oblige le système à se modifier rapidement. Ces systèmes proactifs essaient de dériver une action à la base des informations contextuelles.

L'objectif principal d'un SIP est de fournir des services bien adaptés aux utilisateurs et aux applications. Cette tâche doit se faire selon le contexte actuel et en tenant compte des ressources limitées des équipements. Dans un environnement de services, L'intergiciel est capable d'anticiper les besoins de l'utilisateur et répondre à leurs intentions futures dans un contexte donné, Figure 5.

5 Organisation de la thèse

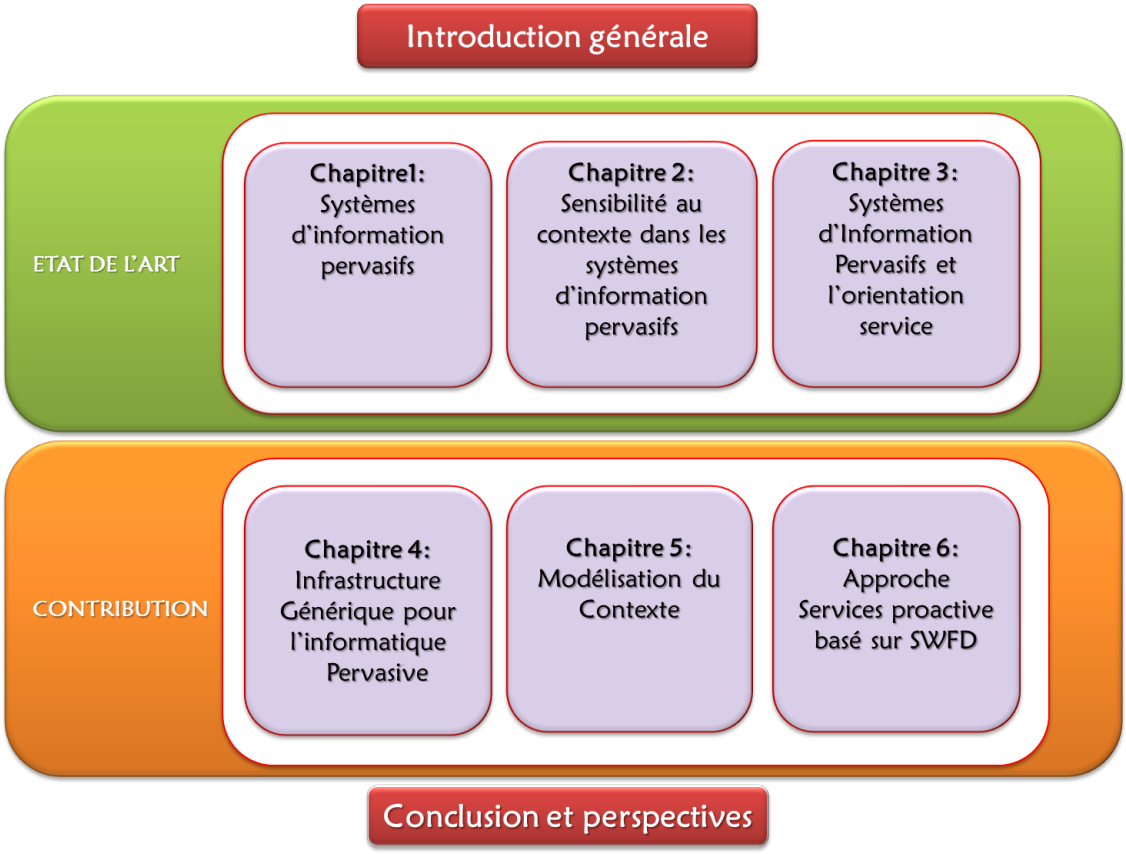

Figure 6 – Plan de Thése.

Ce travail est organisé comme suit :

1. Le premier chapitre, représente un état de l'art sur les systèmes d'information pervasifs. Nous évoquons, dans ce chapitre la définition, l'évolution ainsi que les différents défis et challenges auquels est confrontée cette nouvelle classe de systèmes d'information.
2. Le deuxième chapitre, représente les principaux thèmes liés à notre travail, à savoir le contexte et la sensibilité au contexte. Dans ce chapitre, nous commençons par la définition du contexte, ensuite, nous décrivons les caractéristiques des informations de contexte et les différentes tendances existantes en matière de sensibilité et de modélisation de contexte. Puisque la modélisation du contexte représente l'une des premières phases de prise en compte du contexte dans le processus de création d'applications sensibles au contexte, nous dressons

un état de l'art des travaux de modélisation du contexte, puis nous analysons ces différentes approches pour choisir l'approche la mieux adaptée aux environnements pervasifs.

3. Le troisième chapitre, représente un état de l'art sur l'orientation service dans l'informatique pervasive. Nous commençons par présenter la notion de services et l'émergence des visions techniques. L'orientation service, selon nous, est un concept clé dans la conception des SIP qui doivent faire face à un ensemble de défis. Nous présentons, par la suite, les différents défis auxquels un SIP orienté service doit répondre.
4. Le quatrième chapitre, résume notre solution globale, il présente un cadre conceptuel des SIP, qu'on a appelé "Architecture générique pour l'informatique pervasive " qui tient en compte les éléments et composants majeurs d'un environnement ubiquitaire.
5. Le cinquième chapitre, détaille notre modèle de contexte. Nous utilisons une approche basée sur les ontologies. De plus, nous exposons une approche de raisonnement à base de règles SWRL, pour dériver et inclure des informations contextuelles des services et de l'environnement.
6. Le sixième chapitre, présente le mécanisme proactif de selection de services guidé par le contexte. Dans ce chapitre nous détaillons notre architecture en présentant les différents modules qui permettent d'utiliser les services de façon interopérable, décentralisée et proactive.

   La conclusion résume les principaux apports de nos travaux et les perspectives de recherche que nous entrevoyons pour les prochaines années.

Chapitre I

Les Systèmes D'information Pervasifs

1 Introduction

Le concept de l'informatique pervasive est un terme qui a été introduit pour la première fois au début des années 90 par Mark Weiser au laboratoire Xerox parc [Wei91]. Pour désigner sa vision futuriste de l'informatique du XXIe siècle. Il imaginait un monde peuplé d'objets informatiques et numériques qui seraient reliés en réseaux à très grande échelle et interagiraient de manière autonome et transparente afin d'accomplir diverses tâches de la vie quotidienne. L'informatique pervasive est aussi appelée informatique diffusé, informatique ubiquitaire, intelligence ambiante, informatique omniprésente, ou encore l'internet des objets. Cette dernière appellation souligne le fait que les machines ne communiqueront pas seulement avec les utilisateurs humains, mais elles pourront aussi le faire directement entre elles afin de rendre un service encore plus satisfaisant. Ces constats convergent vers la vision de l'informatique du futur que donnait Mark Weiser en 1991 dans son article fondateur intitulé : <The Computer for the 21st Century> [Wei91] :

"Les technologies les plus profondes sont celles qui disparaissent. Elles s'emmêlent dans le tissu de la vie de tous les jours jusqu'au point où elles en deviennent indiscernables".

Ainsi, pour Weiser [Wei91], trois éléments sont nécessaires : des ordinateurs bon marché et à faible consommation d'énergie qui proposent des interfaces d'affichage plus commodes, des logiciels d'informatique ubiquitaire et un réseau qui relie tous les dispositifs. L'informatique ubiquitaire ne vit pas dans un objet particulier, mais elle est imprégnée dans la structure même de l'environnement qui nous entoure.

L'idée novatrice de la vision de Weiser [Wei91] part du constat que bientôt, chaque personne sera entourée par de nombreux ordinateurs. Nous sommes passés du stade de l'ordinateur central (mainframe) qui a concrétisé l'idée d'un ordinateur pour plusieurs utilisateurs au stade plusieurs ordinateurs enfouis dans des objets de la vie quotidienne pour une même personne. De plus, les communications débordent du cadre classique homme vers homme ou homme vers machine pour inclure des communications directes entre machines, comme l'illustre la Figure 1.1.

La vision de Marc Weiser [Wei91] est devenue aujourd'hui possible, en effet, ces dernières années ont vu la miniaturisation des dispositifs informatiques et électroniques dotés de capacités de capture, de traitement et de communication, pendant que leur puissance de calcul continuait d'augmenter. Beaucoup de progrès ont également été réalisés afin de réduire leur consommation d'énergie. Cette situation a favorisé la démocratisation de ces technologies et leur intégration dans de nombreux objets du quotidien. En outre, le développement des technologies réseaux sans

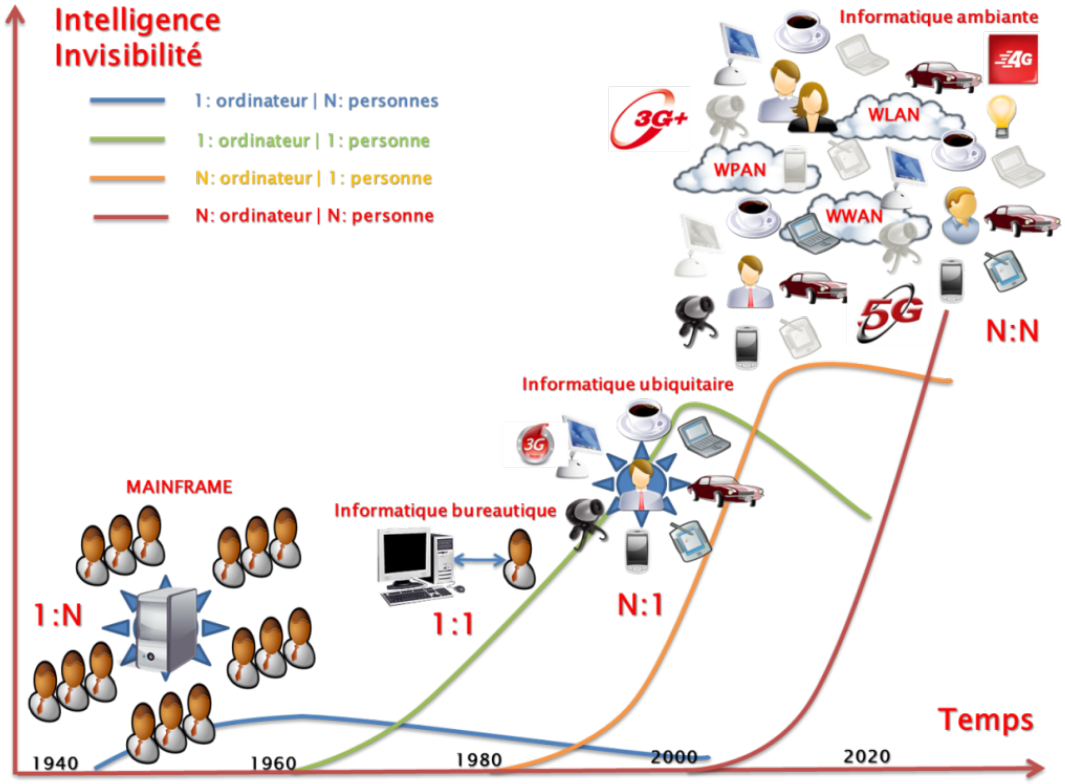

Figure I.1 – Evolution de l'informatique, depuis sa naissance jusqu'à l'informatique ambiante.

fil, Figure 1.2 (Bluetooth, RFID, Wi-Fi, 3G, 4G) a permis à ces nouveaux types d'objets communicants d'interagir spontanément avec l'utilisateur ou avec d'autres objets [Esc08], aussi bien localement qu'à distance, et de favoriser leur mobilité. Ceci se fait d'une manière transparente pour l'utilisateur sans son intervention explicite et lui offre la possibilité de se concentrer sur sa tâche principale au lieu de configurer et de gérer l'ensemble des équipements informatiques mis à sa disposition. En particulier, il s'agit de permettre aux utilisateurs d'accéder aux différents services offerts par ces objets communicants, le plus naturellement possible, n'importe où, à tout instant et à partir de divers dispositifs [BC12], [Sat01].

1.1 Définition des systèmes d'information pervasifs

"Pervasive" est un terme anglais qui signifie omniprésent ou envahissant. Ce mot à été utilisé en informatique pour décrire un monde où l'information et les services sont présents partout et tout le temps. Dans la littérature, ce paradigme a suscité différents axes de recherche plus au moins équivalents : "Informatique Pervasive" (Pervasive computing), "Informatique Ubiquitaire" (Ubiquitous Computing) ou "Intelligence Ambiante" (Ambient Intilligence). Tous ces travaux ont cependant pour principal objectif le partage de l'information, des ressources et des services entre les utilisateurs nomades et leur environnement, où et quand ceux-ci en ont besoin et de manière non intrusive.

L'informatique pervasive a pour origine l'ubiquituous computing de Weiser. Son but est d'intégrer de manière transparente des dispositifs informatiques au monde physique [Wei91]. Elle fait référence à la miniaturisation des dispositifs communicants (processeurs et capteurs) pour une utilisation répandue. L'adjectif ambiant désigne plutôt

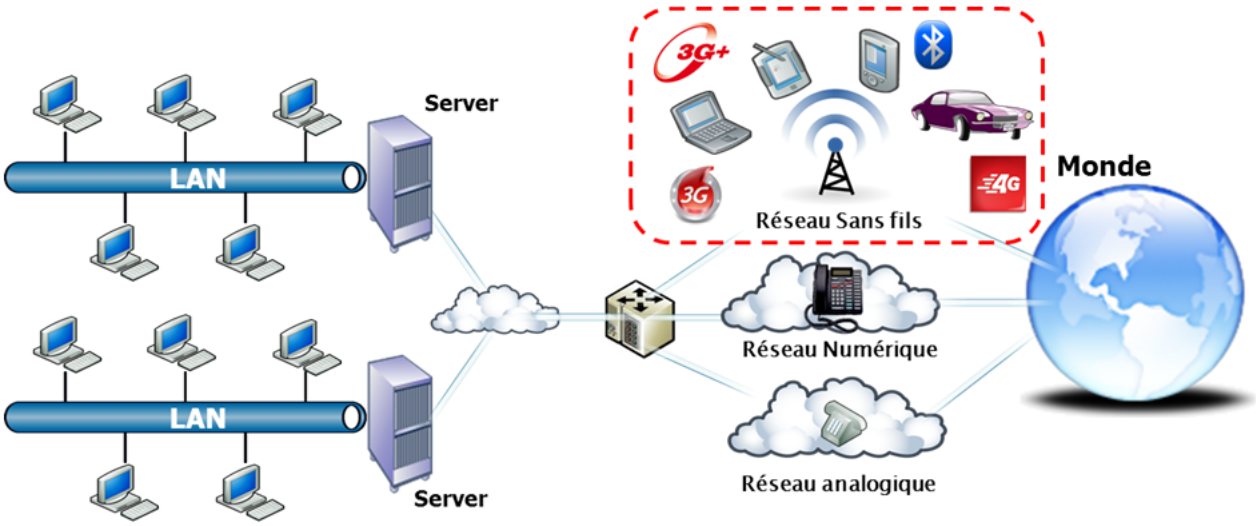

Figure I.2 – Technologie sans fils.

l'environnement artificiel. Dans cet environnement, les réseaux d'unités de calcul évoluent dans le monde réel où les médias de communication et les interfaces tangibles intègrent des objets de la vie quotidienne. La recherche en informatique ambiante constitue alors l'étude de l'évolution logicielle dans un espace ambiant. Il prend en compte l'hétérogénéité des dispositifs. L'informatique diffusé se concentre sur la miniaturisation tout en conservant leurs capacités de mémoire et de calcul.

1.2 Vers des systèmes d'information pervasifs

1.2.1 Évolution des systèmes d'information pervasifs

La vision de Weiser a prouvé son exactitude [Wei91], l'informatique sort de l'ordinateur pour s'intégrer directement dans l'environnement, qui devient donc ubiquitaire. Le but de cette idée est de faire en sorte que l'ordinateur est mis à la disposition de l'utilisateur a tout moment dans cet environnement et lui offre des capacités d'interaction plus naturelles. Les systèmes d'information doivent s'adapter et supporter cette nouvelle vision. Ainsi les objectifs des systèmes informatiques récents ne sont plus restreints à l'exécution de tâches commandées par l'utilisateur mais à faire communiquer plusieurs systèmes mobiles ou fixes pour fournir des services personnalisés. La Figure 1.3 illustre l'évolution de l'informatique vers l'intelligence ambiante en quatre ères :

–L'ère du mainframe (1960-1980) : Dans cet environnement, les utilisateurs accèdent à l'information via un terminal léger disposant d'une application en mode texte, avec de faibles capacités d'interaction, à savoir, les commandes prévues par le serveur. Le concept d'ordinateur correspondait à une machine volumineuse ; la mise en œuvre d'un programme informatique nécessitait un haut niveau d'expertise. L'utilisateur était un lecteur passif de l'information contenue dans un mainframe. En outre, les réseaux étaient du type synchrone avec une connexion physique permanente à un serveur ;

–L'ère de l'ordinateur personnel (1980-2000) : Le paradigme de l'accès à l'information devient celui d'un utilisateur lié à sa machine personnelle de taille réduite pour un coût abordable, contenant des données et programmes personnels et adoptant le schéma d'une architecture client-serveur. Les capacités de connexion via un réseau de

télécommunication sont limitées, car basées sur la logique du réseau téléphonique. Internet existe, son accès reste limité pour le grand public pour des raisons de débit (réseau analogique et essentiellement filaire) ;

–L'ère de l'informatique ubiquitaire (2000-2010) : Apparition d'un nouveau paradigme de l'accès à l'information, un utilisateur muni d'un terminal portatif, ayant de fortes capacités de calcul et de connexion à un réseau sans fil ; la contrainte d'immobilité disparaît donc. D'un point de vue définition d'un système d'information, cela correspond à un utilisateur ayant à sa disposition une infinité de machines informatiques et électroniques dotés de capacités de traitement et de communication. Les applications ne seront plus associées à une machine physique et surtout un écran associe. Elles vont pouvoir migrer et nous suivre au gré de nos déplacements. Les interactions vont être plus naturelles et l'ordinateur va se fondre dans notre environnement et disparaitre ;

–L'ère de l'intelligence ambiante (2010-2020) : L'intelligence ambiante (AmI) est la rencontre de l'informatique ubiquitaire et de l'intelligence. L'objectif est d'exploiter les capacités de perception offertes par tous ces capteurs afin d'analyser l'environnement, les utilisateurs et leurs activités et de permettre au système de réagir en fonction du contexte. La vision de l'AmI est donc toujours celle de l'informatique pervasive, des ordinateurs qui imprègnent l'environnement tout en étant transparents a l'utilisateur, mais en y ajoutant la notion d'intelligence, c'est- à -dire la faculté d'analyse du contexte et l'adaptation dynamique aux situations.

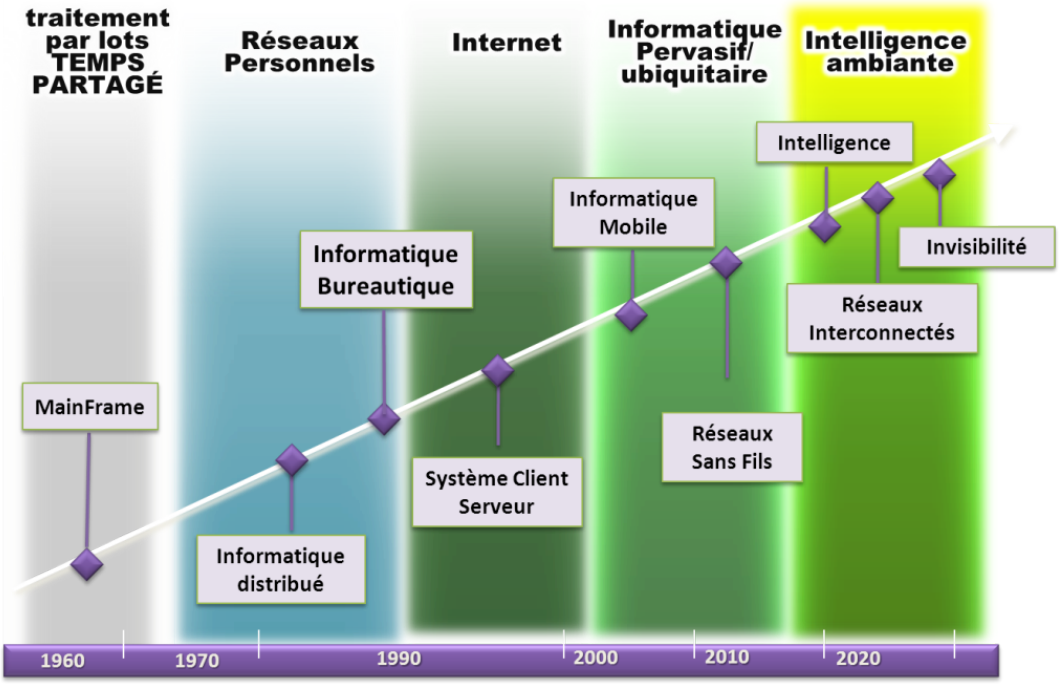

Figure I.3 – Évolution Des systèmes d'informations pervasifs.

1.2.2 Les contraintes des systèmes d'informatique pervasifs

Les SIP constituent une classe émergente de SI dans lesquels les technologies de l'information sont graduellement intégrées à l'environnement physique, de manière à répondre aux besoins et aux envies de à n'importe quel moment. Parmi les principales contraintes de cette nouvelle classe de systèmes d'information, [Table 1.1 : SIP vs SIC] [KB13a] :

1. L'informatique ubiquitaire ayant été conçue dans le but d'être déployée partout et d'être accessible par la

majorité des utilisateurs nomades souvent non spécialistes (citoyen non former). Les dispositifs utilisés ainsi que les logiciels embarqués doivent être maniables et accessibles pour tous.

2. La mobilité des périphériques, le faible débit et l'épuisement d'une batterie entraine des variations de connexion qui peuvent avoir de lourdes conséquences sur le fonctionnement des services (apparition/disparition), l'indisponibilité des services et par conséquent dégradent la qualité de service de l'application.

3. Les systèmes pervasifs se caractérisent par la multiplicité et l'hétérogénéité des dispositifs et d'objets de la vie courante, étant aussi différents que nombreux. Ils sont déployés dans de nombreux domaines d'application, tels que la domotique, la gestion des catastrophes routières, l'armée ou encore les soins hospitaliers. La diversité de ces domaines d'application implique toutes sortes d'objets communicants [AJG12], aussi bien matériels (téléphones portables, caméras vidéo, capteurs) que logiciels (bases de données, agendas en ligne, clients de messagerie instantanée). Ces objets communicants ont des capacités riches et très variées, allant de la détection de mouvements à la réception d'un flux audio/vidéo. De plus, afin de pouvoir coopérer, ils reposent sur des intergiciels spécifiques (CORBA, DCOM, Java RMI, les services Web) et supportent différents protocoles réseaux (SIP, UPnP, X10). Ils peuvent également proposer plusieurs modèles d'interaction. Les informations échangées entre les objets communicants sont elles aussi très hétérogènes. Elles peuvent différer non seulement par leur nature (texte, image, flux audio) mais aussi par leur format (GIF, JPEG ou PNG).

4. L'idéal décrit par Weiser [Wei91] passe par l'invisibilité des machines, or la petitesse de ces éléments entrainent forcement une pénalité dans les ressources de calcul comme la rapidité de traitement, la taille mémoire et la capacité du disque.

5. L'environnement doit offrir une interface personnalisée en fonction du profil de chaque utilisateur. Ceci nécessite une adaptation des services et une disponibilité des ressources en fonction du contexte de l'usager. Ainsi, l'objectif est de répondre au mieux aux attentes des utilisateurs, voire anticiper leurs besoins de manière proactive.

6. Les interactions des systèmes informatiques avec l'extérieur changent de nature : les interactions classiques (principalement des interactions avec les utilisateurs de type clavier/souris) sont remplacées par de nouveaux types d'interactions, notamment des interactions du système informatique avec l'ensemble de son environnement extérieur.

7. Les systèmes pervasifs introduit la notion de sensibilité au contexte en raison de la capacité des objets intelligents de faire la collecte, le traitement et la gestion des informations dans un environnement pervasif. Opposé à l'informatique de bureau, où l'action de l'utilisateur détermine la réaction du système.

2 Les défis et challenges de l'informatique pervasive

Le défi principal de l'informatique ubiquitaire est de fournir un environnement cohérent, impliquant un ensemble d'équipements et de logiciels hétérogènes, distribués et dynamiques, communicant à travers différents protocoles. Dans ce contexte, plusieurs caractéristiques spécifiques au domaine de l'informatique pervasive rendent ce domaine attirant du point de vue des industriels et des utilisateurs. Cette section décrit brièvement chacune de ces caractéristiques [KB14c].

Tableau I.1 – Systèmes d'information pervasif vs Systèmes d'information classique

	Systèmes d'information classique	Systèmes d'information pervasif
Profil utilisateur	Connue Former Employé de bureau	Inconnue Non Former Citoyen
Dispositifs	Ordinateurs personnels	Artefacts multiples
Contexte	Percevoir les données saisit	Percevoir les données du contexte
Services	Services stationnaires	Services mobiles
Tache utilisateur	Specifique	Quelconque
Moyen d'interaction	Localisé homogène Pointer er cliquer	Présence constante Hétérogènes Interfaces multimodales
Espace d'intercation	Cybernitique Fixe	Physique Mobile
Temps utilisation du systeme	Reactive	Proactive

2.1 Technologies liées à l'informatique pervasive :

2.1.1 Les réseaux sans fil

Un réseau sans fil (en anglais Wireless network) représente l'infrastructure de base de tout système pervasif [A.B07], c'est un réseau dans lequel au moins deux dispositifs peuvent communiquer sans liaison filaire. Le périmètre géographique définit l'étendue et la capacité de couverture d'un réseau sans fil. On distingué plusieurs catégories de réseaux sans fil pervasifs, Figure 1.4 :

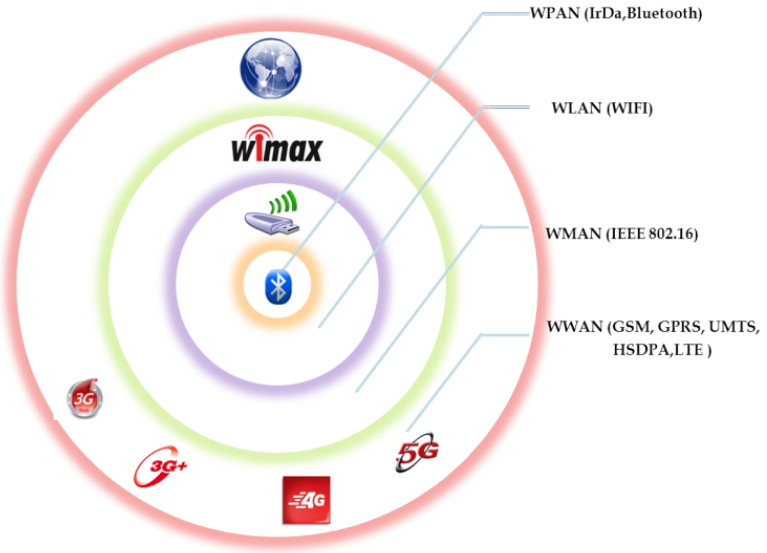

Figure I.4 – types de réseaux pervasifs.

– **Le réseau WPAN pour Wireless Personal Area Network**, concerne les réseaux sans fil d'une faible portée (de l'ordre de quelques dizaines de mètres). Ce type de réseau sert principalement à relier des périphériques (imprimante, appareils domestiques, téléphone portable, capteurs ou un assistant personnel, etc.) à un ordinateur sans liaison

filaire. Plusieurs technologies sont utilisées pour les WPAN dont la principale est la technologie Bluetooth et infrarouge ;

– **Le réseau WLAN pour Wireless Local Area Network**, est un réseau sans fil permettant de couvrir l'équivalent d'un réseau local d'entreprise, soit une portée d'environ une centaine de mètres. Il permet de relier les terminaux présents dans la zone de couverture. Plusieurs technologies concurrentes existent, on peut citer : Le wifi (ou IEEE 802.11) qui offre des débits allant jusqu'à 54Mbps sur une distance de plusieurs centaines de mètres ;

– **Le WMAN pour Wireless Metropolitan Area Network**, est basé sur la norme IEEE 802.16. Il offre un débit utile de 1 à 10 Mbit/s pour une portée de 4 à 10 kilomètres. Cette technologie est destinée principalement aux opérateurs de télécommunication. La norme de réseau métropolitain sans fil la plus connue est le WiMAX, permettant d'obtenir des débits de l'ordre de 70 Mbit/s sur un rayon d'une vingtaine de kilomètres ;

– **Le WWAN pour Wireless Wide Area Network**, est connu sous le nom de réseau cellulaire sans fil. Il s'agit des réseaux de la téléphonie mobile. Plusieurs générations existent :

1G : La première génération de téléphonie mobile possédait un fonctionnement analogique et constituée principalement d'appareils relativement volumineux ;

2G : La seconde génération de téléphonie mobile a été marquée par le passage de l'analogique vers le numérique. Les principaux standards de la téléphonie mobile 2G sont le GSM, CDMA, TDMA. Grâce aux réseaux 2G, il est possible de transmettre la voix ainsi que des données numériques de faible volume (SMS, MMS). Des extensions de la norme GSM (débit maximal de 9,6 kbps) ont été mises au point afin d'en améliorer le débit, c'est le cas notamment du standard GPRS (General Packet Radio System), qui permet d'obtenir des débits théoriques de l'ordre de 114 kbit/s ;

3G : Propose d'atteindre des débits supérieurs à 144 kbit/s, ouvrant ainsi la porte à des usages multimédias tels que la transmission de vidéo, la visio-conférence ou l'accès à internet haut débit. La principale norme 3G utilisée s'appelle UMTS (Universal Mobile Telecommunications System) qui permet le transfert de la voix et de données avec des débits pouvant aller de 384 kbps à 2 Mbps ;

3.5 G : La technologie HSDPA (High-Speed Downlink Packet Access) est un protocole de téléphonie mobile de troisième génération baptisé " 3.5G " permettant d'atteindre des débits de l'ordre de 8 à 10 Mbits/s ;

4G : est la 4e génération des standards pour la téléphonie mobile. Elle permet le " très haut débit mobile ", c'est-à-dire des transmissions de données à des débits théoriques supérieurs à 100 Mb/s, voire supérieurs à 1 Gb/s. Les débits sont en pratique de l'ordre de quelques dizaines de Mb/s selon le nombre d'utilisateurs, puisque la bande passante est partagée entre les terminaux actifs des utilisateurs présents dans une même cellule radio ;

5G : Samsung Electronics, a récemment testé avec succès un réseau de téléphonie mobile à la norme 5G transmettant des données à la vitesse de 1 Gbps sur une distance de deux kilomètres séparant deux terminaux alors que les réseaux 4G actuels transmettent à 75 Mbps. L'exploitation de ces nouveaux réseaux très hauts débits devraient avoir lieu en 2020 au plus tôt. Il faudra donc patienter pour pouvoir atteindre des vitesses de transmission permettant de télécharger un film en une seconde, " Les utilisateurs bénéficieront d'un large éventail de services comme les films en 3D, mais aussi les jeux et la lecture en streaming (direct) de contenu en ultra haute-définition (UHD) ". La nouvelle technologie utilise des bandes à extrêmement haute fréquence (EHF).

2.2 Les réseaux de capteurs sans fil

La tendance de la miniaturisation du matériel informatique a apporté une nouvelle génération de réseaux informatiques et télécoms présentant des défis importants [Kac09]. Les réseaux de capteurs sont l'une des technologies de

l'informatique embarquée ayant la possibilité de traitement, de stockage, de détection et de communication à petite échelle et à faible coût. Les réseaux de capteurs constituent une catégorie de réseaux sans fil comportant d'un très grand nombre de noeuds, avec un déploiement très dense et à grande échelle dans des environnements pervasifs. Ces noeuds sont utilisés pour l'acquisition de données et leur transmission à une station de traitement appelée : Station de Base. Les capteurs d'aujourd'hui relèvent plusieurs informations, les communiquent entre eux, et même analyse des données, Figure 1.5.

Plusieurs applications envisagées dans les réseaux de capteurs font toujours l'objet d'une recherche universitaire ou industriel. On trouve des applications pour la détection et la surveillance des désastres, le contrôle de l'environnement et la cartographie de la biodiversité, le bâtiment intelligent, l'agriculture de précision, la surveillance et la maintenance préventive des machines, la médecine et la santé, la logistique et la télématique, etc.

2.2.1 Architecture d'un noeud capteur

Un noeud-capteur est composé de plusieurs modules correspondant chacun à une tâche particulière [Kac09] :

L'unité d'acquisition des données : il s'agit de répondre à une variation des conditions d'environnement par une variation de certaines caractéristiques électriques (par exemple une variation de température entraîne une variation de la résistance. Les variations de tension sont ensuite converties par un convertisseur Analogique-Numérique afin de pouvoir être traitées par l'unité de traitement.

L'unité de traitement des données : les microcontrôleurs utilisés dans le cadre de réseaux de capteurs sont à faible consommation d'énergie. Leurs fréquences sont assez faibles, moins de 10 MHz pour une consommation de l'ordre de 1 mW. Outre le traitement des données, le microcontrôleur commande également toutes les autres unités notamment le système de transmission.

L'unité de transmission de données : La quantité d'énergie nécessaire à la transmission augmente avec la distance. Pour les réseaux sans fil classiques (LAN, GSM) la consommation d'énergie est de l'ordre de plusieurs centaines de milliwatts, alors que pour les réseaux de capteurs, le système de transmission consomme environ 20 mW et possède une portée de quelques dizaines de mètres. Pour augmenter ces distances tout en préservant l'énergie, le réseau utilise un routage multi-sauts.

La source d'énergie : pour des réseaux de capteurs sans fils autonomes, l'alimentation est une composante cruciale. Il y a essentiellement deux aspects : premièrement, stocker l'énergie et la fournir sous la forme requise ; deuxièmement, tenter de reconstituer l'énergie consommée par un réapprovisionnement grâce à une source externe au noeud-capteur telles les cellules solaires. Le stockage de l'énergie se fait traditionnellement en utilisant des piles.

2.2.2 Architecture d'un réseau de capteurs

Un réseau de capteurs est généralement constitué de nombreux noeuds répartis dans une zone de couverture (sensor field)[Kac09]. Ces noeus doivent surveiller certains phénomènes grâce à leurs capteurs puis envoient les informations à un puits. Le puits est un noeud particulier doté d'une puissance de calcul supérieure et d'une quantité d'énergie potentiellement infinie. Ce puits peut être connecté à Internet ou possède un lien radio de type GSM ou GPRS qui lui permet d'envoyer les informations (données ou alertes) à un centre de contrôle pour l'utilisateur final. Il peut y avoir plusieurs puits mobiles ou fixes dans un réseau, Figure 1.5).

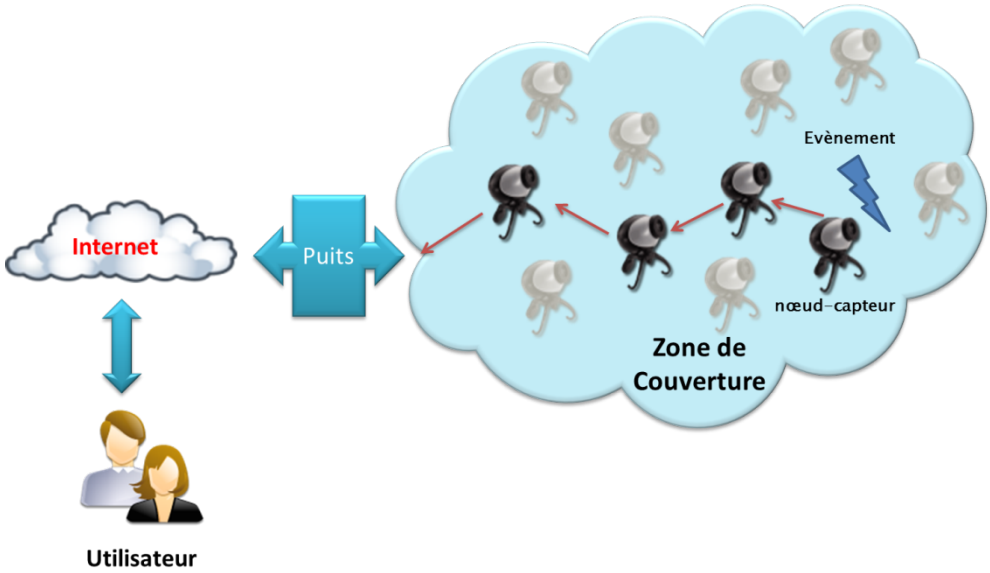

Figure I.5 – Architecture d'un réseau de capteurs.

2.3 Dispositif à accès Pervasif (Objets communicants)

A la base d'un environnement ambiant, les objets communicants qui le forment. Ils s'apprêtaient alors à envahir notre environnement privé, public et professionnel. Ils sont nés dans le milieu des années 90 grâce à l'émergence des nouveaux moyens de communication sans fil et de la miniaturisation des composants et des technologies de traitement de l'information [Sen07], [MS03]. Un objet communicant peut être défini comme une entité physique capable de percevoir et de communiquer avec son environnement, avec des utilisateurs et avec d'autres objets quelconques interagissant avec lui.

Figure I.6 – Objet communicant : Aibo7.

Par conséquent, l'objet communicant ne peut accomplir cet objectif que s'il a les capacités suivantes :

- **Mémorisation** : stockage des informations caractéristiques de l'objet et de son environnement ;
- **Perception** : surveillance de son environnement physique et informatique dans le but de s'adapter et de gérer

son évolution ;

– **Communication** : recherche et sélection d'information auprès de son environnement ;

– **Action** : traitement d'information et exécution des mécanismes afin d'accomplir une tâche précise ;

– **Décision** : analyse de son état, du comportement de son environnement pour prendre de façon autonome et/ou concertée des décisions relatives à des situations particulières.

On voit de plus en plus d'objets communicants faire leur apparition sur le marché et dans la vie quotidienne [Zai09]. Parmi ces objets l'on peut citer Aibo7, Figure 1.6, le chien robot de Sony qui est équipé de plusieurs capteurs tels qu'une caméra et un microphone, qui est relie à internet via le réseau wifi et qui est doté d'algorithmes de vision par ordinateur et de reconnaissance vocale qui lui permettent, dans une certaine mesure, d'interagir avec son environnement.

Figure I.7 – Objet communicant : Nabaztag.

La vision d'Olivier Mevel et RaHaladjian correspond à un espace dans lequel la plupart des objets qui nous entourent seraient doués d'intelligence, capables de réagir ou d'interagir avec nous et surtout et toujours connectés au réseau [Zai09]. Cette vision correspond à celle de Mark Weiser [Wei91], le père de l'informatique ubiquitaire. Il s'agit de la première fois que cette vision passe la frontière entre le monde de la recherche et le monde commercial du grand public. Le premier objet de cette marque qui apparaissait sur le marché est : le Nabaztag Figure 1.7, un lapin connecté à internet et communicant par des lumières, de la synthèse vocale ou le mouvement de ses oreilles ; un appareil qui peut fonctionner sans écran et sans clavier. Il diffuse des informations du type météo, bourse, circulation routière, arrivée de courriers.

2.4 Plasticité des Interfaces homme machine

Selon la vision de Weiser [Wei91], les objets de la vie quotidienne deviennent des supports d'interaction. Ainsi, les dispositifs d'interaction se diversifient par leur forme et leur finalité : les ordinateurs, les TV, les téléphones, les PDA, etc. Ces dispositifs supportent des fonctionnalités de plus en plus nombreuses et utilisent des techniques intelligentes d'interaction. De nos jours, la flexibilité et la plasticité des IHM deviennent une nécessité avec l'émergence de nouvelles technologies d'affichage et d'interaction qui peuvent s'intégrer dans l'environnement ambiant.

Le domaine de l'interaction homme-machine (IHM) consiste à étudier les moyens d'interactions entre les technologies développées et les humains, [DFAB97], [Gal12]. Les systèmes pervasifs proposent de remplacer les interfaces d'interactions traditionnelles comme le clavier ou la souris par d'autres modes d'interaction plus intuitifs. La nouvelle

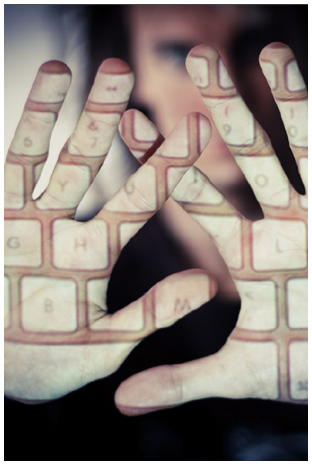

Figure I.8 – Interfaces Pervasifs.

ère des interactions des utilisateurs qui peuvent impliquer la participation des gestes, tactile multiple, commande vocale, le contrôle du regard, etc. Ces interfaces s'adaptent en fonction du contexte auxquelles elles sont soumises. Ainsi, La sensibilité au contexte peut être utilisée pour adapter des interfaces homme-machine. La manipulation et la gestion du contexte intéressent les chercheurs en IHM pour construire des interfaces plastiques, c'est-à-dire des interfaces adaptées au contexte.

La miniaturisation des dispositifs d'interaction facilite la mobilité des supports d'interaction et ainsi la possibilité d'interagir avec le système à partir de lieux différents. Par conséquent, le système doit reconnaître ces différents lieux et adapter ses paramètres en fonction des conditions contextuelles (en réunion, en voiture, chez soi). De plus, aujourd'hui, il est possible d'utiliser des systèmes interactifs pour une communication plus intelligente en déduisant des commandes avec des perceptions gestuelles, corporelles ou encore faciales. Par exemple, une interface de service météo présente à l'utilisateur la météo de l'endroit où il se trouve, en le localisant de manière automatique et transparente.

2.5 La Gestion d'énergie

La multiplication des technologies d'accès offre aujourd'hui une multitude de solutions réseaux, offrant chacune des caractéristiques et des services différents [Lou12]. Le WiFi par exemple, est de plus en plus disponible autour de nous, maison, lieu de notre travail, hôtels, aéroports et administrations. Cependant, la technologie Wi-Fi ne garantit encore ni la gestion de la mobilité ni la qualité de Service. Les réseaux mobiles, dits aussi cellulaires offrent plus de garanties en termes de qualité de Service et de mobilité. Outre leurs différences en termes de performances et de services offerts, les différentes technologies d'accès possèdent des modèles de consommation d'énergie différents. Ces différences sont intrinsèquement liées aux technologies utilisées, il résulte des différences au niveau de la consommation d'énergie. Or, pour un terminal mobile qui se base sur une batterie de faible capacité, pourra etre indisponible pour un temps indéfini. Cet aspect est primordial puisqu'il affecte directement la disponibilité des services pour l'utilisateur. Il est donc important de faire en sorte qu'une charge de batterie, dure un maximum de temps. Les limitations en termes de taille et poids suggèrent que les concepteurs de systèmes pervasifs devraient évaluer et sélectionner les sources alternatives d'énergie.

2.6 Propriétés indispensables des systèmes d'informations pervasifs :

Les besoins engendrés par Internet et la prolifération d'objets communicants entrainent de nouveaux défis et propriétés indispensables [KB14c] :

2.6.1 la distribution

Un système réparti est un ensemble d'ordinateurs (téléphones mobiles, PDA, capteurs...) indépendants qui apparaît à un utilisateur comme un seul système cohérent. Les ordinateurs peuvent être regroupés dans un même lieu ou dispersés dans l'environnement physique et sont accessibles à travers différents protocoles [Pet10]. Cette coordination se fait le plus souvent par envoi de messages via un réseau de communication qui peut être un LAN (Local Area Network), WAN (Wide Area Network) et Internet [Bou08]. La communication est transparente et invisible à utilisateur [TvS08].

[Li10], [Dug07], definissent un système distribué comme etant plusieurs ordinateurs autonomes géographiquement éloignés qui communiquent à travers un réseau informatique. Ces dispositifs interagissent les uns avec les autres pour réaliser une tâche courante s'exécutant en général en parallèle sur différents noeuds matériels. Nous pouvons modéliser ces systèmes par des réseaux d'unités de calcul et de stockage de données possédant la capacité de communiquer entre eux [Figure 8]. le but est de coordonner l'utilisation des ressources partagées pour fournir des services à leurs utilisateurs. Les systèmes distribués peuvent être classés de différentes manières [DSAM]. Trois classes ont été essentiellement identifiées à savoir les systèmes de calcul distribué (Distributed Computing Systems), les systèmes d'information distribués (Distributed Information Systems) et les systèmes pervasifs distribués (Distributed Pervasive Systems).

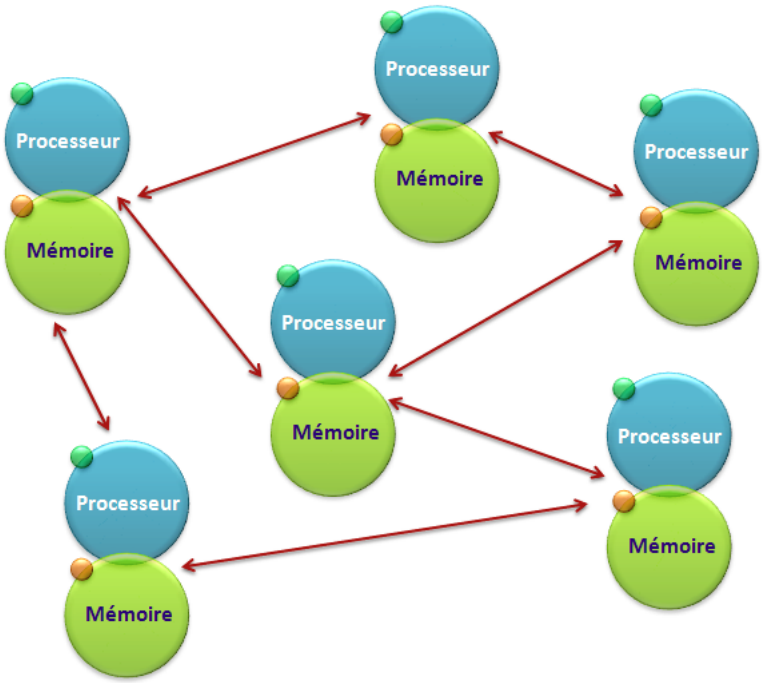

Figure I.9 – Système Distribué.

Les systèmes distribusé sont considérés comme le noyau des sytemes pervasifs, en effet ils peuvent fonctionner

dans des environnements homogènes et peuvent également prendre en charge des nouveaux contextes et ressources. Dans de tels systèmes, les dispositifs d'informatique omniprésente sont liés utilisant un réseau sans fil. De nouveaux dispositifs peuvent être introduits et rendus accessibles et d'autre anciens peuvent être modifies et retités.

2.6.2 La mobilité

La mobilité est un atout pour les systémes pervasifs [Lou10]. L'apparition des dispositifs mobiles comme les PC portables, les PDA et les téléphones mobiles, ainsi que des réseaux sans-fil renda la mobilité un des challenges les plus importants de l'informatique pervasive [Saa09]. La mobilité est le tout premier attribut que peut posséder un système informatique portable. [PR04a], [Hoa07], [Rei10],[Kou05] difinissent la mobilité : les équipements pouvant être transportés par leurs utilisateurs et représentant la possibilité de changer la localisation. Dans ce sens, un changement de l'environnement spatial d'une entité peut influencer le comportement, la structure de l'application et entraine des variations de connexion qui peuvent avoir de lourdes conséquences sur le fonctionnement des services. Un équipement peut donc ne pas toujours être accessible à tout moment. L'informatique mobile doit prendre en considération la localisation [TLR+09], les dispositifs mobiles sont en effet utilisés dans des situations de la vie de tous les jours. Les services qu'ils doivent rendre sont souvent en rapport avec l'activité de l'utilisateur.

La mobilité réfère à la capacité d'accéder à des services ou à des applications indépendamment de la localisation physique des utilisateurs, de leurs comportements ou de leurs mouvements [PR04b]. L'informatique mobile se distingue principalement de l'informatique classique fixe par le nomadisme, c.-à-d. la mobilité des utilisateurs et de leurs équipements, aussi bien du côté du matériel (la limitation de la taille des claviers et de l'écran, de la bande passante, de l'énergie disponible) que du côté de l'utilisateur (qui a une attention limitée ou peut avoir les deux mains occupées, par exemple). La mobilité impose de nombreuses contraintes de conception concernant la taille, le poids, les capacités de traitement et d'affichage des unités mobiles, la largeur de bande sans-fil et les déconnexions intermittentes, l'autonomie des batteries, les problèmes de la dissipation de chaleur, etc.

2.6.3 Interoperabilité

L'interopérabilité en générale est la capacité de deux ou plusieurs systémes ou components d'échanger informations et d'utiliser les informations échangées, même avec des différents langages d'implémentation [VHT00]. L'interopérabilité est plus que jamais cruciale dans un environement pervasif, cela permet de définir la manière dont des applications ou des systèmes communiquent et échangent. Cette notion a été centrale dans le développement des réseaux de communication.

2.6.4 La scalabilité(passage à l'échelle)

En informatique et en télécommunication, le passage à l'échelle est une propriété très importante pour un système [DRW06]. Il se réfère à sa capacité à faire face à une augmentation du travail à effectuer et à s'adapter au rythme de la demande [NKPSS12], [Sar10]. Le nombre d'équipement présent dans un environnement pervasif peut être très important, ce nombre engendre le problème de passage à l'échelle des applications s'exécutant dans ce type d'environnement [Cos08]. Il faut donc que les systèmes pervasifs soient capables d'appréhender un grand nombre d'équipements qui de plus sont dynamiques [Bou08]. Indisponsable dans les systèmes pervasifs, la scalabilité permet de déceler les limitations du système lors d'une montée en charge le nombre de composants entrant en jeu est très important [Esc08].

Le passage à l'échelle est nécessaire car le nombre d'interactions dans les espaces intelligents fait que l'utilisation des ressources (par exemple l'énergie, la bande passante ou l'attention de l'utilisateur) devient critique [San10]. Les études existantes sur le passage à l'échelle ne considèrent pas la localisation des entités ; par exemple dans un serveur web, on essaie de servir le plus de clients possibles indépendamment de leur situation géographique. Dans le cas des systèmes ubiquitaires, la plupart des interactions ont lieu entre des entités proches, ce qui peut être exploité pour améliorer le passage à l'échelle. En effet, quand un utilisateur s'éloigne de l'environnement ubiquitaire, on pourra réduire le nombre d'interactions qu'il a avec les éléments de cet environnement, car elles seront moins pertinentes.

2.6.5 Hétérogénéité

Par rapport aux premiers systèmes distribués, les systèmes ubiquitaires sont composés d'objets communicants hétérogènes à plusieurs niveaux matérielles et logicielles, [DRW06], [Jou09]. Il existe actuellement un grand nombre de technologies logicielles et de protocoles de communication pour le domaine de l'informatique pervasive [Bou08]. Les équipements utilisés sont très variés : un utilisateur peut se servir d'un ordinateur portable, d'un PDA ou encore d'un smart phone [FGM05],[FLRT08], [Lou10]. Ces équipements fonctionnent avec des systèmes d'exploitation variés (tels que Windows, Linux, Windows CE, PalmOS, Symbian, etc.). Du point de vue du réseau sans fil, différentes technologies sont à la disposition de l'utilisateur : Wi- Fi (IEEE 802.11a/b/g) ou Bluetooth pour des communications courtes/moyennes portée, GPRS ou UMTS pour une couverturenationale voire internationale. Au sein d'un même réseau, des stations de travail aux capacités de calcul et de stockage importantes peuvent coexister avec d'autres appareils à faibles ressources comme des assistants numériques personnels [CFW09], [Hoa07].

2.6.6 Integration

Beaucoup d'éléments de l'informatique existent déjà et sont déployés dans des environnements réels [SM03], il faut essayer de les integrer avec les nouveaux objets que propose l'informatique pervasive pour créer un environnement ubiquitaire. Plus ces éléments sont nombreux, et plus l'intégration devient complexe. Il faut prendre en compte des aspects tels que la qualité de service, la sécurité, l'invisibilité et la fiabilité.

2.6.7 Dynamique

Dans un système d'informatique ubiquitaire, la disponibilité des objets communicants peut varier au cours du temps [NKPSS12]. En effet, ces derniers peuvent être dynamiquement ajoutés ou retirés du système, de manière intentionnelle ou non. Par exemple, la portée limitée des technologies réseaux sans fil, associée à la mobilité des utilisateurs, peut placer hors de portée certains objets communicants. De plus, ces objets communicants ont généralement des ressources limitées en énergie et sont sujets à des défaillances matérielles et logicielles, ce qui peut les rendre indisponibles pour un temps indéfini. Le système doit alors s'adapter en découvrant de nouveaux objets communicants afin de poursuivre son exécution et potentiellement reprendre certaines tâches interrompues.

2.6.8 Autonomie

L'intervention de l'homme dans le contrôle et la maintenance des systèmes informatiques tend à disparaître au profit de systèmes capables de s'autogérer [en.12]. Ces systèmes sont capables d'agir sur leur propre fonctionnement afin de s'adapter aussi bien à des conditions d'exécution changeantes qu'à une erreur survenue en leur sein [Sar10]. L'objectif de définir de tels systèmes est d'alléger la tâche du concepteur, du déployeur et du mainteneur d'applications

qui ne peuvent pour des systèmes de grande complexité, appréhender toutes les reconfigurations ou adaptations qui sont susceptibles d'intervenir dans le cycle de vie des logiciels.

Nous pouvons citer l'initiative pour l'autonomic computing, lancée par IBM, qui définit des systèmes autonomes [BBC+03] comme des systèmes capables de :

1. **S'auto-configurer (self-configuration)** : cette propriété caractérise la capacité d'un système autonome à prendre en compte dans sa configuration actuelle et celle du réseau le supportant, de nouveaux services et l'arrivée d'un nouvel équipement ; aucune procédure d'installation manuelle n'étant nécessaire ;
2. **S'auto-réparer (self-healing)** : le système est capable de détecter, diagnostiquer et d'effectuer les réparations nécessaires suites à différentes pannes (e. g. crash d'une machine) ;
3. **S'auto-optimiser (self-optimization)** : le système est capable de redistribuer les différentes tâches sur les différentes machines composant le réseau afin de garantir une utilisation optimale des ressources ;
4. **s'auto-protéger (Self-protection)** : le système est capable de détecter et de se protéger des attaques (e.g. virus, hackers) éventuelles.

2.6.9 La pro-action

A pour but de prévoir les services et données qui seront nécessaires ; c'est une forme d'adaptation prévisionnelle. La pro-action a pour objectif de préparer le traitement d'une demande utilisateur avant même que cette demande ne soit explicite [Laf07]. Cela nécessite une historisation des situations et actions, une analyse des situations et actions antérieures et leur rapprochement avec la session actuelle.

2.6.10 L'invisibilité

Est une contrainte forte ; il ne faut pas encombrer l'utilisateur avec des considérations qui ne le concernent pas, il faut qu'il se concentre sur la tâche à réaliser [BBC97]. Le concept d'invisibilité est directement repris des travaux de Mark Weiser [Wei91], qui imaginait que l'informatique allait disparaître peu à peu pour s'intégrer à l'environnement (technologie calme). Il incite l'informatique à ne pas submerger les utilisateurs d'informations, mais plutôt à devenir un outil invisible [Liu06]. Le but est d'obtenir une distraction minimale des utilisateurs pour une meilleure productivité et satisfaction. En effet, avec l'augmentation du nombre de dispositifs informatisés autour des utilisateurs, on assiste aussi à une augmentation du nombre d'applications qu'ils doivent gérer.

2.6.11 Sensibilité au Contexte

Le développement de l'informatique sensible au contexte constitue l'un des développements majeurs de la vision de Weiser [Wei91]. Le but est de concevoir des systèmes capables de recueillir et de traiter des informations sur l'environnement (les informations du contexte) afin d'y adapter leur comportement pendant l'exécution. L'idée principale de l'informatique sensible au contexte est de sortir le plus possible les humains de la boucle de contrôle des machines, en réduisant les interactions entre l'humain et la machine. [SAW94] furent les premiers à proposer une définition de la sensibilité au contexte : il s'agit de la capacité d'un système à découvrir et à réagir à des changements dans l'environnement où il se trouve. Ils signalent également l'importance de l'adaptation du système à ces changements.

Dans une définition similaire, [Bro96] définit le contexte comme les éléments de l'environnement de l'utilisateur que l'ordinateur de l'utilisateur connaît. [Pas98] définit le contexte comme sous-ensemble des états physiques et conceptuels. [Jam01] ajoute le comportement de l'utilisateur et des interactions actuelles avec le système pervasive.

[VLA01] soulignent l'importance de capteurs intégrés dans l'environnement afin de détecter l'emplacement et le mouvement actuel de l'utilisateur et l'ajouter aux propriétés d'un système sensible au contexte.

La prise en compte du contexte d'utilisation dans les applications est un domaine de recherche d'actualité connu sous le nom de " sensibilité au contexte " (ou context-awareness en anglais) [ADB+99]. Une application sensible au contexte doit percevoir la situation de l'utilisateur dans son environnement et adapter par conséquent son comportement à la situation en question. La première action prise par un système généralisé, le système perçoit de l'information contextuelle à partir de capteurs multiples récupère et traite les informations contextuelles et les présente sous une forme appropriée à l'utilisateur final. La perception (ou sensibilité au contexte) est nécessaire afin de fournir un service satisfaisant et proactif aux utilisateurs [CV09]. Cependant, elle introduit beaucoup de complexité. Par exemple, il faut pouvoir gérer l'incertitude, fusionner des données fournies par des capteurs différents (et qui parfois seront contradictoires), traiter ces informations en temps réel, etc.

2.6.12 Gestion du Contexte

Les systèmes ubiquitaires permettent l'automatisation de tâches auparavant attribuées aux utilisateurs. Ces systèmes ubiquitaires doivent être sensibles à leur environnement physique pour prendre les décisions appropriées. Ils doivent alors non seulement prendre en compte la dynamicité des objets communicants, mais aussi l'évolution des paramètres physiques de l'environnement physique. Ces paramètres physiques sont les phénomènes observables de l'environnement physique, par exemple, la température ambiante, la lumière du soleil ou la présence d'un utilisateur. Ils sont généralement mesurés par des objets communicants appelés capteurs.

L'analyse de ces paramètres constitue le contexte des environnements. Par exemple, des valeurs élevées des paramètres physiques de température et de fumée peuvent constituer un contexte d'incendie. [Dey01] définit le contexte comme l'ensemble des informations permettant de caractériser la situation des utilisateurs, des lieux ou des objets. Les systèmes ubiquitaires analysent les changements de contexte pour décider du comportement à adopter et ainsi s'adapter. La nature du contexte est directement liée au domaine applicatif cible. Le contexte donne aux systèmes ubiquitaires une vision haut niveau et dirigée de l'évolution des environnements physiques. Un environnement physique est ainsi abstrait pour former un environnement ubiquitaire. Ce dernier est constitué de services fournis par les objets communicants et du contexte nécessaire à l'adaptation des applications.

2.6.13 Adaptabilité

Les systèmes pervasifs sont constamment confrontés à l'évolution de leur contexte d'exécution. Ces évolutions peuvent comprendre les modifications du comportement, de la localisation aussi bien que les changements de comportement ou de disponibilité des autres logiciels [Bou08]. Les applications s'exécutant sur ce type d'environnement doivent donc être capables de s'adapter à ces changements et mettre en place des stratégies pour pallier aux différents événements pouvant intervenir lors de leur exécution. Dans les systèmes d'information pervasifs, on doit assurer une adaptation pour garantir une utilisation confortable des applications dans ces nouveaux environnements. Pour la réaliser, beaucoup de nouveaux paramètres entrent en jeu :

1. **paramètres réseau** : dans les réseaux sans fil la bande passante est limitée, les connexions sont intermittentes, la qualité de service n'est pas évaluée de la même façon ;
2. **paramètres de l'utilisateur** : Des contraintes d'utilisabilité et d'ergonomie se présentent aux concepteurs de ce genre d'application. Ainsi, on doit prendre en considération ses préférences, son emplacement géographique, son profil ;

3. **paramètres du terminal** : la diversité des terminaux mobiles influe sur la conception de ces systèmes d'information. Le comportement de ces systèmes doit s'adapter aux capacités matérielles et logicielles de ces appareils.

Tous ces paramètres forment des contextes d'utilisation différents. Dans la plupart des cas, ces paramètres n'ont pas été pris en compte lorsque l'application a été développée. Ceci conduit généralement les informaticiens à reprendre leur cycle de vie dès son début pour prendre en compte ces nouveaux paramètres.

2.6.14 Sécurité et confidentialité

Une des caractéristiques importantes des systèmes pervasifs est la simplicité d'utilisation. L'informatique ubiquitaire ayant été conçue dans le but d'être déployée partout et d'être accessible par la majorité des utilisateurs (souvent non spécialistes)[Saa09]. Les dispositifs utilisés ainsi que les logiciels embarqués doivent être maniables et accessibles pour tous. Cependant, la simplicité attendue peut présenter un défi important surtout quand il s'agit de maintenir un équilibre entre accessibilité et sécurité. Cette mobilité de l'utilisateur et la multiplication des dispositifs légers et autonomes accentuent et rendent plus complexes les problèmes de sécurité. En effet, la mise en place d'environnements pervasifs permet à l'utilisateur nomade de solliciter ou de communiquer avec d'autres utilisateurs, des ressources ou des services au sein d'environnements ou de domaines qui lui sont inconnus. Les données contextuelles sont en général des données personnelles qui représentent des données statiques sur l'utilisateur comme ses préférences et ses habitudes et des données dynamiques comme sa localisation et les tâches qu'il effectue. Ainsi, ces données possèdent un caractère confidentiel et l'échange de ces données entre applications doit respecter les mesures de sécurité dé nies par l'utilisateur. Un système est sécurisé lorsqu'il est capable de résister à des utilisations non autorisées tout en fonctionnant nominalement [Esc08]. La sécurité peut s'évaluer suivant six critères différents :

1. **La confidentialité** : Est le fait de garantir que des informations confidentielles ou privées ne seront pas révélées à des tiers qui ne doivent pas en avoir connaissance ;
2. **L'intégrité** : a pour but de garantir qu'un système est protégé contre les dysfonctionnements, les agressions et les attaques. Plus particulièrement, l'intégrité doit assurer que les données ne soivent pas corrompues de manière volontaire ou accidentelle ;
3. **La disponibilité** : vise à garantir qu'un système doit rester disponible malgré les potentielles attaques ou utilisations non autorisées. Ce critère est devenu primordial. En effet de nombreuses applications sont distribuées et la non-disponibilité d'un seul des noeuds peut mettre en péril toute l'application ;
4. **L'authentification** : tend à assurer à un utilisateur utilisant des données de l'identité de l'émetteur des données qu'il consulte. Par exemple, une application traitant des données issues d'un réseau de capteurs RFID doit pouvoir vérifier que les données sont bien issues de ces capteurs et n'ont pas été rajoutées ;
5. **La non-répudiation** : vise à empêcher que l'auteur d'une donnée puisse prétendre ensuite qu'il n'en est pas l'auteur ; elle implique l'intégrité, mais s'étend au-delà, car doit apporter la preuve que des informations ont bien été émises par l'auteur.
6. **La gestion des preuves** : doit fournir un système archivant toutes les transactions du système afin de pouvoir vérifier l'authenticité des données échangées.

2.6.15 Service pour l'informatique ambiante

L'environnement ambiant est constitué de l'ensemble des services accessibles et des dispositifs utilisables. Cette infrastructure de gestion de services d'objets communicants permet de découvrir, de composer et d'interagir avec des services hétérogènes.

1. **Découverte de services** :La découverte de services est un élément clé de l'informatique ubiquitaire, le défi consiste à informer un utilisateur entrant dans une zone précise de l'existence de services et améliore son utilisation de façon transparente [Bro96]. Les systèmes ubiquitaires utilisent des mécanismes de découverte de services afin de localiser dynamiquement les objets communicants dont ils ont besoin [ZMN05]. La découverte de services permet aux développeurs de ne pas avoir à connaître où sont physiquement localisés les objets communicants. Cette découverte peut être centralisée ou distribuée. Dans le premier cas, la découverte se fait à l'aide d'un annuaire qui centralise les descriptions des services et les adresses physiques associées des objets communicants. Dans le second cas, tous les objets communicants participent à la découverte de services, soit en envoyant périodiquement la description des services qu'ils fournissent, soit en envoyant des requêtes décrivant les caractéristiques des services dont ils ont besoin. Il apparaît donc important de pouvoir spécifier clairement quels types de services sont disponibles dans un environnement, afin de les mettre à disposition des différents systèmes déployés.

2. **Composition de Services** : Dans un environnement de communication ambiante, une application coordonne des fonctionnalités issues d'objets communicants variés. Une composition de services a pour but de définir comment ces fonctionnalités interagissent afin de réaliser le comportement attendu pour l'application.

3. **Exécution de Services** : L'exécution de services permet, à partir de la composition, de remplir une tâche utilisateur. Le Système Pervasif, du fait de son hétérogénéité, rend complexe l'interopérabilité. Les différents services à composer peuvent avoir différentes interfaces et modes de communications. L'exécution en elle-même doit se faire de manière optimisée pour prendre en compte des ressources limitées (réseau, énergie). Le Système Pervasif apporte un côté quasi temps-réel incitant à des temps de réponse réduits.

4. **Gestion de la qualité de service** : Il n'existe pas de consensus autour de la définition de la qualité de service. Néanmoins il existe un standard X.902 défini par l'International Télécommunications Union [idn96] qui décrit la qualité de service comme un ensemble d'exigences concernant le comportement collectif d'un ou de plusieurs objets. L'objectif est de proposer aux utilisateurs des applications qui fonctionnent le mieux possible et le plus longtemps possible sur leur appareil mobile favori et ce quelles que soient les évolutions du contexte [Lou10].

A présent qu'internet est à la portée de tout un chacun, les applications se sont multipliées et par conséquent la notion de qualité de service est de plus en plus utilisée à d'autres fins que l'évaluation des réseaux. Les définitions que l'on retrouve le plus souvent dans la littérature s'accordent à dire qu'actuellement il n'est plus possible d'évaluer la qualité de service d'une application en tenant compte uniquement des critères réseaux et matériels, Alors que la plupart des travaux se concentrent uniquement sur un ou deux aspects de la qualité de service, l'utilisateur et l'environnement de l'application influencent la qualité de service liée à l'utilisation. L'infrastructure regroupe les périphériques, les composants logiciels et le réseau. Au niveau de l'infrastructure, nous souhaitons garantir à la fois la continuité de service et une durée de vie la plus longue possible de l'application malgré les variations des ressources matérielles, logicielles et réseaux et malgré la mobilité des périphériques.

3 Conclusion

Dans cette premiere partie, Nous avons effectué une analyse complète du domaine des systèmes d'informatique ubiquitaire. À partir des résultats de cette analyse nous avons dressé une liste de défis et chalenges. Parmi les principales caractéristiques de cette nouvelle classe de SI, [KG06] observe que, au-delà de l'hétérogénéité des dispositifs, les SIP doivent également offrir de la sensibilité au contexte, issue de la possibilité de collecter, d'analyser et de gérer les données environnementales grâce aux dispositifs utilisés. Nous étudierons dans le chapitre deux l'intérêt de la sensibilité au contexte dans un environnement ambiant.

Chapitre II

La Sensibilite au contexte dans les systemes d'information pervasifs

1 Introduction

Comme annoncé par Weiser en 1991 [Wei91], les systèmes pervasifs permettant de doter tout objet de la vie quotidienne de capacité de calcul et de communication, et rendant son intelligence transparente à l'utilisateur, est une réalité de nos jours. Ainsi, les systèmes d'information pervasifs sont nés de l'ambition de fournir un accès au système d'information en tout lieu et à tout moment avec un contenu correspondant à la situation de l'utilisateur. Contrairement au poste de travail traditionnel, l'informatique ubiquitaire se caractérise par des changements constants de l'environnement. Le plus souvent, ces changements sont dus à la mobilité de l'utilisateur. Dans le domaine de l'informatique ubiquitaire, une classe d'applications dites sensibles au contexte a suscité un grand intérêt en particulier depuis l'émergence des technologies sans fils.

L'étude de la sensibilité au contexte dans les systèmes d'information pervasifs nous amène tout d'abord à étudier les travaux de recherche effectués pour définir et modéliser le contexte. La modélisation permet d'offrir les fondations pour une représentation expressive du contexte et de simplifier son utilisation [BB06]. Les modèles de description du contexte varient selon leur expressivité et le type d'informations de contexte qu'ils permettent de décrire. Cette variété de modélisation provient de l'utilisation du contexte dans les domaines variés de l'ingénierie comme le traitement du langage, les systèmes d'aide à la prise de décision ou les systèmes sensibles au contexte.

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

Dans ce chapitre, nous commençons notre étude par la définition du contexte, ensuite, nous décrivons les caractéristiques des informations de contexte et les différentes méthodes d'acquisition de ces informations. Puisque la modélisation du contexte représente l'une des premières phases de prise en compte du contexte dans le processus de création d'applications sensibles au contexte, nous dressons un état de l'art des travaux de modélisation du contexte, puis nous analysons ces différentes approches pour choisir l'approche la mieux adaptée aux environnements pervasifs.

1.1 Motivation

En voici un scénario pour motiver l'informatique pervasive réactive au contexte : Considérons un hôpital intelligent où des patients, des infirmières, des médecins, etc. sont impliqués [Dej07]. Supposons que l'hôpital est équipé de technologies (matériel et logiciel) de capteur de contexte dans les chambres, les couloirs et le jardin à la disposition des individus impliqués. Un système de l'informatique pervasive ré au contexte adapté pour la surveillance et le suivi des patients dans les hôpitaux aide à minimiser l'engagement des spécialistes aux activités moins importantes.

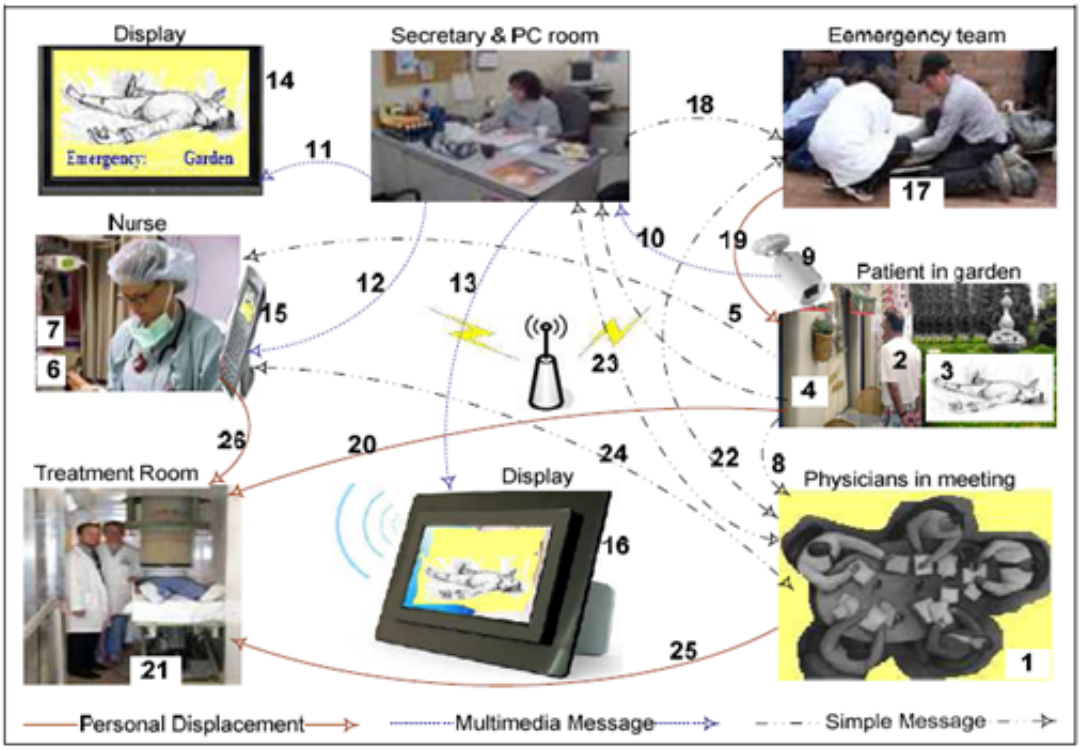

Figure II.1 – scénario pour motiver l'informatique pervasive réactive au contexte : hôpital intelligent.

L'intervention humaine pourrait être nécessaire uniquement lors d'une alerte par le système. La Figure 2.1, illustre un cas d'utilisation spécifique du scénario de l'hôpital intelligent [Dej07]. Un matin, le docteur Amine et ses collègues sont en réunion de consultation hebdomadaire (1). Karim, le patient, est dans un jardin pour profiter du soleil matinal (2). Soudain, Karim se sent épuisé et tombe par terre (3). Les portables et les insignes qu'il porte remet immédiatement toutes les informations nécessaires au dispositif informatique se trouvant à proximité (4). Ensuite le système envoie (5) un message d'alerte à Rafik (6), l'infirmier, sur son téléphone intelligent (7). Sachant les horaires du docteur Amine à partir de son agenda, qu'il est dans une réunion, le système lui informe (8) par message SMS sur son téléphone cellulaire une lumière rouge lors de sa réunion. La camera face au jardin (9), où Karim est situé, est activé et son image est envoyée (10) à l'ordinateur central pour l'adapter et la diffuser (11, 12, 13) à tous les terminaux concernés (14, 15, 16). Les secouristes (17) sont également informés (18) de la situation. À la suite, Karim est amené (19, 20) à la chambre de traitement (21) par les secouristes d'urgence. Le docteur Amine, qui a déjà été au courant de la situation actuelle de son patient, a fait toutes les consultations nécessaires (22, 23, 24), la préparation de médicaments appropriés et il est déjà dans la salle de traitement (25). Rafik est également dans la

salle de traitement (26) pour fournir l'aide nécessaire au patient. Ce travail est donc motivé par deux observations principales que nous avons relevées au sujet de l'environnement de l'informatique pervasive et le système réactif au contexte : la nécessité d'un modèle de gestion du contexte dans un cadre générique et la nécessité d'une plateforme pour développement des systèmes adaptés au contexte dans l'informatique pervasive.

2 Notion de contexte

2.1 Définitions

La définition du terme contexte reste vague du fait de son utilisation dans plusieurs domaines. Avant de présenter les définitions du contexte proposées dans le domaine de l'informatique pervasive, nous allons donner les définitions fournies par quelques dictionnaires de référence.

Que disent les dictionnaires ?

–**Le Petit Robert :** ensemble des circonstances dans lesquelles s'insère un fait.

–**L'encyclopédie Larousse :** ensemble des conditions naturelles, sociales, culturelles dans lesquelles se situe un énoncé, un discours ; ou encore : ensemble des circonstances dans lesquelles se produit un événement, se situe une action.

–**Hachette Multimédia :** ensemble des éléments qui entourent un fait et permettent de le comprendre.

–**Le Grand dictionnaire numérique en ligne :** Le contexte est un ensemble d'informations concernant l'action du stylet, en rapport principalement avec sa localisation à l'écran, qui permet au système d'exploitation de l'ordinateur à stylet de différencier les commandes et l'entrée des données, et de fonctionner en conséquence.

D'après ces définitions, un contexte est un ensemble d'éléments ou de circonstances qu'on peut assimiler à des informations qui entourent. Dans le domaine informatique, il existe deux types de définition du contexte : le premier type définit le contexte en énumérant des entités spécifiques qui représentent le contexte associé à une application, comme la localisation par exemple, alors que le second type définit le contexte d'une manière conceptuelle et se focalise sur la structure des informations de contexte (une approche formelle). Pour faire suite à notre recherche, nous allons présenter dans l'ordre chronologique d'apparition une liste non exhaustive des définitions du contexte dans le domaine de l'informatique. Plusieurs travaux définissent le contexte à travers des énumérations ; [CK00] en fait une synthèse intéressante. Parmi ces travaux :

– Schilit, Adams et Want, 1994 [SAW94] ont considéré que le contexte possède trois aspects importants qui consistent en des réponses aux questions suivantes : Où es-tu ? Avec qui es-tu ? De quelles ressources disposes-tu à proximité ;

– Schilit et Theimer [SAW94] dans leur projet AMS (Active Map Service) considèrent le contexte comme un ensemble d'informations de localisation et l'identité des personnes et des objets à proximité ainsi que les modifications pouvant intervenir sur ces objets ;

– Brown [Bro96]a défini le contexte comme : les éléments de l'environnement d'un utilisateur dont l'ordinateur à connaissance ;

– Brown, Bovey et Chen[BBC97], ces derniers considèrent le contexte comme étant un ensemble d'informations de localisation, de profil de personnes auxquels ils ajoutent des informations temporelles et l'orientation spatiale (direction) ;

– Ryan et al [RPM97] ajoutent à cette énumération des informations sur l'environnement telles que la localisation de l'utilisateur, l'environnement, l'identité et le temps ;

– Ward et Hopper [WJH97], les états des environnements possibles de l'application ;

– Pascoe [Pas98] : ensemble d'états physiques et conceptuels ayant un intérêt pour une entité particulière.

Dans le second type de définitions plus générales du contexte, on trouve les définitions suivantes :

– Schmidt et al [SAT+99] ont dégagé les éléments suivants pour décrire le contexte :

– **Contexte du dispositif :** processeurs disponibles, dispositifs accessibles pour l'interaction input/output, capacité de réseau, connectivité, coût de dispositifs ;

– **Contexte de l'utilisateur :** localisation, personnes à proximité, situation sociale ;

– **Contexte physique :** niveau de bruit et de luminosité.

– Brézillon et Pomerol [BcP99] définissent le contexte comme étant toutes les connaissances qui participent à la résolution d'un problème. En intelligence artificielle, Brézillon [BcP99] définit le contexte par : "Le contexte est ce qui n'intervient pas directement dans la résolution d'un problème mais contraint sa résolution" ;

– Chen et Kotz, 2000 [CK00] : ensemble des états environnementaux et paramètres qui déterminent le comportement d'une application ou dans lequel un événement de l'application se déroule et ayant un intérêt pour l'utilisateur.

– La définition la plus générale et la plus utilisée est celle donnée par [Dey00] :

Toute information pouvant être utilisée pour caractériser la situation d'une entité (Personne, objet physique ou informatique). Et plus généralement tout element pouvant influencer le comportement d'une application.

Cette définition permet de prendre en compte toutes les entités (abstraites ou concrètes) considérées comme pertinentes lors d'une interaction entre un utilisateur et une application. Elle permet de couvrir une large gamme d'applications sensibles au contexte. Cette définition a été réduite dans [DAS01] afin de se focaliser sur une classe d'applications. Les auteurs identifient alors quatre catégories de contextes considérés comme les plus utilisés d'un point de vue pratique : la localisation, le profil de l'utilisateur, l'activité et enfin le temps.

– Henricksen, Indulska et Rakotonirainy [HI04] : la circonstance ou la situation dans laquelle une tâche informatique se déroule.

Malgré le grand nombre de définitions existantes et les similarités (la plupart font référence à la localisation et l'environnement), le mot contexte reste toujours général. En ce qui nous concerne, nous utilisons la définition donnée par Dey [DAS01] pour le reste de ce document. Un contexte observable associé à une entité peut être capturé par le système ou calculé à partir d'autres données (interprété). Dans le cadre de cette thèse, le contexte est utilisé par des applications dans le but d'extraire les informations pertinentes du contexte afin de proposer des services adaptés.

2.2 Importance du contexte dans le domaine informatique sensible au contexte

Dans le domaine de l'informatique pervasive ou ambiante, la notion de contexte regroupe les applications ou les systèmes qui utilisent le contexte pour adapter leurs comportements. En se basant sur les travaux de Dey et Abowd [ADB+99], Schilit et al [SAW94] et de Brown et al [BBC97], Chalmers a identifié six utilisations possibles du contexte dans un environnement sensible au contexte [CDS04] :

– Présentation et affichage des informations de contexte comme par exemple la localisation.

– Association d'informations de contexte à une donnée de l'application, comme exemple l'association des per-

sonnes présentes à une conférence et leur localisation au programme de la conférence,

– La configuration sensible au contexte, comme par exemple le fait d'effectuer une impression sur l'imprimante la plus proche, sans intervention de l'utilisateur,

– Lancement d'actions de réaction selon la valeur du contexte, comme le chargement d'une carte géographique d'une zone quand l'utilisateur se déplace vers cette zone,

– Médiation contextuelle, qui consiste par exemple à utiliser le contexte pour modifier un service fourni.

– Présentation sensible au contexte qui consiste à modifier l'interface d'une application selon le contexte.

2.3 Caractéristiques des informations de contexte

Les informations de contexte sont des informations collectées à partir de plusieurs sources hétérogènes [BB06]. De ce fait, elles ont des caractéristiques variables. Chaque contexte observable peut être statique ou dynamique.

–**Les observables statiques** : représentent des informations qui ne changent pas au cours du temps. Il suffit de les collecter au lancement de l'application. Exemple : la taille d'un écran et le profil d'un utilisateur.

–**Les observables dynamiques** : représentent des informations dont les valeurs changent fréquemment. Les observations de contexte effectuées à partir de capteurs physiques peuvent être sujets à des bruitages ou des erreurs de capture [BB06]. Ainsi, ces observations sont incorrectes lorsqu'elles ne reflètent pas l'état réel de l'environnement, incohérentes lorsqu'elles contiennent des informations contradictoires et incomplètes lorsque certains aspects du contexte ne sont pas renseignés.

2.4 Catégories de contexte

Etant donné la diversité des informations composant le contexte, il est utile d'essayer de les classer par catégorie pour faciliter leur utilisation [Cha07]. Les entités principales concernées par la notion de contexte sont des lieux, des personnes ou des objets. Les lieux sont des régions d'espaces géographiques comme des chambres, des bureaux, des bâtiments, des rues ou des zones bien définies. Les personnes peuvent êtres des individus ou des groupes d'individus rassemblés ou répartis. Les objets peuvent êtres des entités physiques, des composants logiciels ou des artefacts (applications, fichiers, ressources). Nous classons les informations contextuelles utilisées dans la majorité des travaux existants en quatre catégories principales :

L'identité : se réfère à la capacité d'assigner un unique identifiant à une entité. Cet identifiant doit être unique dans l'espace de nommage utilisé par les applications.

La localisation : ne se limite pas à la position 2D des objets et des personnes. En effet, elle peut concerner l'orientation, l'altitude et les relations spatiales entre les entités (comme les relations de proximité, de coexistence et de contenance). La localisation peut référencer des lieux. Ces lieux peuvent être identifiés par leurs emplacements géographiques absolus ou relatifs par rapport à des objets de référence.

état ou activité : encapsule les caractéristiques des entités qui interviennent dans le système. Par exemple, pour un lieu, l'état peut caractériser la température ambiante, la quantité de lumière existante ou le niveau de bruit courant. Pour une personne, l'état peut se référer à ses signes vitaux, sa fatigue ou son activité (par exemple, il est en train de conduire, lire, marcher, courir). Pour les composants logiciels, des exemples de ces caractéristiques sont le temps de réponse d'un service, son taux d'utilisation, son état (disponible, activé, désactivé).

Le temps : il peut caractériser une entité. Le temps permet d'établir un historique de valeurs permettant d'enrichir le contexte. En effet, l'enchaînement et l'ordonnancement d'actions ou d'évènements dans le temps peuvent aussi être importants pour la décision prise par l'application.

Les données encapsulées par ces quatre catégories peuvent être interprétées ou corrélées pour obtenir des informations contextuelles supplémentaires afin de garantir une évaluation plus étendue d'une situation. Par exemple, en connaissant dans une salle, le nombre de personnes, leurs positions relatives et l'amplitude du bruit, on peut déterminer si elles sont en conférence ou non.

3 Notion de Sensibilité Au Contexte (Context-Awareness)

3.1 Introduction

Depuis quelques années, les avancées sur la mobilité des périphériques et sur la technologie sans fil ont ouvert de nouvelles perspectives dans divers domaines de recherche liés à la communication [Lou10]. Alors que traditionnellement, les applications se contentaient de produire de nouvelles données en sortie à partir de données en entrée, aujourd'hui les traitements peuvent dépendre également de la situation géographique de l'utilisateur, de ses souhaits ou encore des données physiques telles que la température extérieure ou le taux de luminosité. Ces données annexes font partie d'un ensemble appelé données contextuelles ou contexte de l'application. L'utilisateur n'est pas le seul à percevoir l'influence du contexte sur le service rendu. Ces définitions montrent qu'à présent les applications doivent intégrer des mécanismes d'acquisition du contexte permettant de le prendre en compte afin de s'adapter par des reconfigurations pour répondre au mieux aux besoins du service rendu.

La caractéristique principale d'un systéme informatique pervasif est le changement dynamique de l'environnement ou bien plus précisément du contexte. Pour mieux aider l'utilisateur dans ces tâches quotidiennes, les systèmes informatiques pervasifs doivent tenir compte du contexte global et adapter leurs services aux utilisateurs selon ce dernier. Cette aptitude est connue sous le nom de sensibilité au contexte. Ce terme a été évoqué pour la première fois par [SAW94] dans leurs travaux sur un système de localisation. Ils ont défini la sensibilité au contexte comme l'aptitude d'une application à s'adapter au contexte de son exécution selon : La localisation, l'ensemble des personnes à proximité et les machines. [Bro96] a défini la sensibilité au contexte dans son travail relatif à un guide touristique comme toute application qui prend en compte le contexte de l'utilisateur. [DAS01] l'a défini comme un système qui utilise le contexte pour fournir des informations et/ou des services pertinents à l'utilisateur. Selon lui la pertinence dépend de la tâche de l'utilisateur.

Ces définitions sont en effet très proches et s'articulent autour de l'aptitude d'un équipement informatique à changer ou à adapter son comportement en se basant sur le contexte de l'utilisateur, l'environnement et de l'application. Le principe architectural classique de la figure 2.2 permettant la prise en compte du contexte. Les informations contextuelles proviennent le plus souvent de différentes sources. Ces informations sont recueillies et utilisees par la plate-forme afin d'adapter le comportement et la structure des services.

3.2 Définitions et objectifs des systèmes sensibles au contexte

Un système est dit sensible au contexte s'il peut récupérer, interpréter et utiliser des informations caractérisant le contexte et adapter sa réponse en fonction de ces informations. On distingue clairement aujourd'hui entre les

Figure II.2 – Architecture en couche des applications sensibles au contexte.

applications qui utilisent le contexte de manière statique, d'autres adaptent dynamiquement leur comportement en fonction du contexte.

– [SAW94] définissent un système sensible au contexte comme tout système qui s'adapte selon son endroit d'utilisation, la collection de personnes et objets voisins aussi bien que le changement de ces objets.

– [NBR97] considèrent la sensibilité au contexte comme la capacité d'un système à capturer, interpréter et réagir à un aspect changeant de l'environnement d'un utilisateur et des dispositifs qu'il utilise.

– Selon Dey [Dey00], il stipule qu'un système est sensible au contexte s'il utilise le contexte pour fournir des informations pertinentes et/ou des services à l'utilisateur, cette pertinence des données dépend des tâches de l'utilisateur.

– Xiaohang [Xia03] a constaté que les définitions des systèmes sensibles au contexte ignorent la description des procédures de gestion contexte dans un environnement sensible au contexte. De ce fait, il a donné une nouvelle définition qui considère que la sensibilité au contexte d'un système logiciel est sa capacité à acquérir, gérer, interpréter et répondre aux changements du contexte afin de fournir les services appropriés.

– Informatique sensible au contexte ou Context Aware Computing : terme introduit par Schilit and Want, [SAW94]

– Brown et al. 1997, [BBC97] : Les applications sensibles au contexte sont des applications dont le comportement peut varier en fonction du contexte.

– Dey et al. 2001, [DAS01] : Un système est sensible au contexte s'il utilise le contexte pour offrir des informations ou des services pertinents pour l'utilisateur.

– En couplant ces deux définitions : Les applications sensibles au contexte sont des applications dont la structure et le comportement varient en fonction du contexte. Elles utilisent les observations de contexte pour offrir des informations ou des services pertinents pour l'utilisateur.

Comme il a été observé et discuté dans plusieurs travaux Chaari [Cha07], Keidl et Kemper, [AFG+04], un système sensible au contexte a plusieurs composants : capteur de contexte, stockage du contexte, raisonneur du contexte et consommateur du contexte.

3.3 Sensibilité au contexte pour quelle informatique ?

Informatique mobile : (mobilité utilisateur, mobilité du terminal, mobilité reseau) caracterisée par un changement de contexte permanent.

Informatique ubiquitaire : informatique accessible de tout endroit et depuis toute sorte de dispositif.

Informatique diffusé : à pour but d'offrir des services aux utilisateurs interagissant avec leur environnement (y compris les dispositifs). [Sat01].

Informatique ambiante : Milieu ayant la faculté de percevoir, de raisonner, d'agir, et d'interagir afin de fournir des services ameliorant la qualité de vie des personnes.[CC08].

4 Architecture d'un systéme sensible au contexte

Dans le domaine de l'informatique pervasive, un système sensible au contexte est caractérisé par des fonctionnalités qui lui permettent d'acquérir des informations de contexte en interagissant avec des capteurs, d'interpréter ces données et de les analyser pour détecter les changements pertinents qu'ils subissent [BB06], et enfin, de s'auto-adapter ou d'adapter lors de la détection des situations pertinentes. La conception des applications sensibles au contexte soulève de nouveaux défis :

- les méthodes classiques de développement des logiciels sont difficiles à appliquer sur les applications sensibles au contexte ;
- les solutions proposées pour ce genre d'applications sont soit spécifiques à un besoin précis soit elles manquent d'abstraction lors de leur conception ;
- la capture du contexte est souvent distribuée et mène à des conceptions complexes.

Dey [DAS01] a été le premier à proposer une séparation entre l'acquisition de contexte et son utilisation dans les applications. Il pensa qu'il est utile de séparer la gestion du contexte de l'application, afin de pouvoir développer une plateforme générique de développement et déploiement d'applications sensibles au contexte . Cette idée a été reprise par la majorité des travaux actuels dans le domaine de la sensibilité au contexte, proposant ainsi des architectures à plusieurs couches [Cha07]. Nous remarquons que toutes ces propositions se basent sur cinq couches principales : capture du contexte, interprétation/agrégation du contexte, stockage/historique du contexte, dissémination du contexte et application, Figure 2.3.

4.1 Couche Acquisition du contexte

L'acquisition du contexte est un processus principal lors de la conception d'un système sensible au contexte [Sou10]. Ce processus vise à permettre au système de capturer les informations contextuelles nécessaires à son fonctionnement. Pour ce faire, deux manières peuvent être utilisées : l'acquisition explicite et/ou l'acquisition implicite.

- Acquisition explicite : Cette acquisition permet à l'utilisateur d'exprimer lui-même les informations contextuelles dont le système a besoin pour lui restituer la réponse la plus pertinente. Par exemple, quand l'utilisateur lance une requête, il peut exprimer ses caractéristiques statiques et ses préférences mais aussi sa localisation, etc.
- Acquisition implicite : les informations contextuelles sont déterminées sans l'intervention de l'utilisateur par des matériels et/ou des logiciels spécifiques pour ce but. Des paramètres du contexte sont récupérés implicitement par des capteurs. Par exemple le type de dispositif utilisé via un programme java, la localisation de l'utilisateur

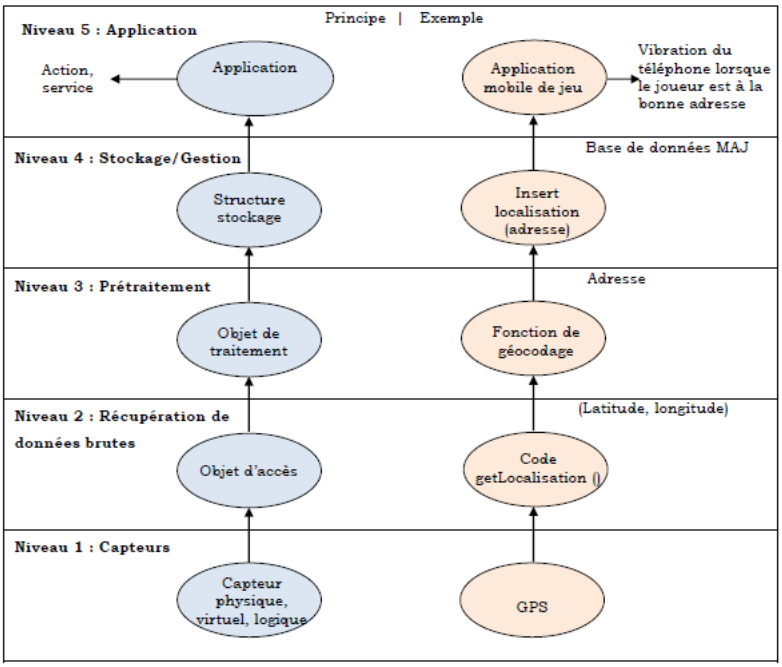

Figure II.3 – Architecture d'un systéme sensible au contexte.

via un capteur physique de géo-localisation GPS, la détection de la température ou du niveau de bruit dans une salle à l'aide d'un capteur physique disposé dans cette salle.

L'acquisition des informations de contexte peut se faire selon plusieurs méthodes, indépendamment de l'application. Mostefaoui a classé ces méthodes en trois catégories [MPRB04] :

4.1.1 Acquisition par profil

Cette méthode consiste à récupérer des informations soit à travers une interface graphique soit par l'intermédiaire d'un fichier de profil ;

4.1.2 Acquisition par sonde

Cette méthode consiste à utiliser des sondes (capteurs) pour récupérer les informations de contexte. On distingue deux types de sondes : les sondes physiques et les sondes logiques. Les sondes physiques sont des composants électroniques qui permettent de mesurer des paramètres physiques dans l'environnement, par exemple l'utilisation de GPS pour la découverte de la localisation d'un utilisateur ou encore l'utilisation de capteurs de mouvement pour détecter la présence de quelqu'un dans un espace (une salle), la détection de la température ou du niveau de bruit dans une salle à l'aide d'un capteur physique disposé dans cette salle, alors que les sondes logiques utilisent des logiciels pour récupérer des informations associées à l'hôte où s'exécute l'application. Par exemple il est possible de détecter l'emplacement d'un livreur de marchandise en consultant son carnet électronique de rendez-vous sans avoir recours à des capteurs physiques. Nous pouvons aussi détecter l'activité de l'utilisateur sur un micro-ordinateur en analysant les évènements de la souris ou les saisies à partir du clavier. Les capteurs virtuels sont beaucoup moins couteux que les capteurs physiques puisqu'ils sont basés sur des composants logiciels ;

4.1.3 Acquisition par dérivation

La dérivation ou l'interprétation du contexte consiste à utiliser un ou plusieurs observables pour déduire ou calculer à la volée un contexte de plus haut niveau en utilisant des méthodes d'interprétation. Par exemple, la rue où se trouve une personne est un contexte de haut niveau calculé à partir des données de localisation en latitude et longitude collectées par un capteur GPS, Figure 2.3.

L'acquisition du contexte à partir de capteurs n'est pas une tâche facile pour un développeur d'applications, cela est dû au fait que chaque capteur a une API propre et qu'un même observable peut être collecté à partir de capteurs différents et peut nécessiter un traitement supplémentaire (interprétation) avant qu'il ne puisse être utilisé par une application. La collecte du contexte peut être menée de plusieurs façons :

- Par un accès direct de l'application à la source du contexte.
- En utilisant une infrastructure intergicielle qui interagit avec la source du contexte, et qui prend en compte certains aspects liés à la gestion du contexte comme son interprétation par exemple.

La première alternative nécessite que l'application s'interface directement à la source du contexte pour le collecter. Le problème de cette méthode réside dans le fait que le code d'interaction de l'application avec cette source du contexte est noyé dans le code fonctionnel de l'application, ce qui ne permet pas de réutiliser ce code pour d'autres applications qui interagissent avec la même source de contexte. De plus, le changement de la source du contexte nécessite la réécriture de l'application. La deuxième alternative consiste à utiliser une infrastructure intergiciels pour interagir avec les sources du contexte vise à séparer la collecte du contexte de l'application. Cette séparation a pour but de cacher à l'application les détails se rapportant aux sources du contexte.

4.2 Couche interprétation du contexte

L'interprétation du contexte représente le mécanisme qui déduit un contexte de haut niveau en utilisant d'autres informations de contexte [BB06]. Les informations utilisées pour déduire un contexte de haut niveau peuvent être des informations interprétées, des informations capturées ou des informations de profil. À chaque fois qu'une nouvelle donnée est collectée, le mécanisme d'interprétation doit être relancé pour déduire les nouvelles valeurs du contexte.

En effet, les capteurs fournissent généralement des données techniques qui ne sont pas appropriées pour une utilisation directe par l'application [Cha07]. Les transformations effectuées sur les données brutes fournies par les capteurs peuvent être réalisées par plusieurs opérations : extraction, quantification, raisonnement et agrégation. Par exemple, les coordonnées GPS d'une personne peuvent être moins significatives qu'une adresse physique sous forme de numéro de rue et de ville. La complexité des interprétations de contexte peut varier d'une simple agrégation de valeurs qui proviennent de plusieurs capteurs à des raisonnements ou analyses statistiques complexes. Par exemple, la localisation de plusieurs personnes dans une seule salle peut inférer le fait qu'ils sont en réunion. Dans ce cas, le niveau de bruit peut aussi être une information importante pour savoir s'ils sont en réunion de travail ou de loisir.

Cette couche doit aussi assurer la résolution de conflits causés par l'utilisation de plusieurs sources de contexte. En effet, ces sources peuvent donner des résultats contradictoires ou peuvent aboutir à des situations imprécises. Cette couche doit donc avoir une certaine intelligence d'interprétation pour résoudre ces conflits.

4.3 Couche de stockage et historique du contexte

La troisième couche stockage et historique du contexte organise les données capturées et interprétées et les stocke pour une utilisation ultérieure. Ce stockage peut être centralisé ou distribué [Cha07]. La solution centralisée est l'option

la plus répandue et la plus utilisée puisqu'elle facilite la gestion des mises à jours et des variations des valeurs du contexte. La gestion distribuée du contexte est beaucoup plus complexe puisqu'elle inflige des fonctions additionnelles de découvertes de ressources et d'actualisation des valeurs du contexte. De plus, cette gestion distribuée alourdit la tâche de l'application qui doit gérer la collecte des différentes informations contextuelles d'une façon interne. Pour stocker une information, nous avons besoin de définir un modèle pour la décrire. Ainsi, un modèle de contexte est requis pour pouvoir l'utiliser dans l'application.

4.4 La modélisation du contexte

Pour décrire la sensibilité d'une application à son contexte d'exécution, il faut déterminer les contextes auxquels cette application est sensible et les décrire dans un modèle [BB06]. Par conséquent, la modélisation du contexte est la première étape dans le processus de création d'applications sensibles au contexte. Cette modélisation permet à l'application de faciliter l'interaction avec le contexte en fournissant une description abstraite des observables. La diversité des informations de contexte et leur utilisation dans divers domaines engendrent différentes façons de les modéliser. Nous allons passer en revue ces approches représentatives de modélisation de contexte et quelques exemples correspondants.

4.4.1 Approche paires/triplets

Les approches paires/triplets permettent de modéliser les valeurs observées du contexte ; elles utilisent à cet effet des structures de données très simples pour les décrire. Les premiers travaux visant à décrire le contexte sont ceux de [AV12]. Ils proposent d'utiliser un serveur d'environnement dynamique qui se charge d'observer un ensemble de contextes pour le compte d'une application. Chaque observation du contexte est sauvegardée dans une variable d'environnement constituée d'une paire clés/valeurs. l'exemple illutsre une variable d'environnement qui permet de décrire les imprimantes les plus proches.

Nearest Printer=HP :HP4 :HP6

Schmidt [Sch10] modélise le contexte comme un triplet constitué de l'observable, de la valeur observée du contexte, et un degré de certitude sur la cohérence des données observées. L'exemple decrit de contexte à l'aide de ce triplet.

C= (Location, MeetingRoom, 95)

L'approche paires/triplets, et plus particulièrement celle qui utilise la modélisation clés/valeur, est fréquemment utilisés pour décrire des informations de localisation.

4.4.2 Approche orientée modèle

L'approche orientée modèle utilise des modèles formels pour modéliser les informations de contexte. Son objectif principal est d'offrir la possibilité d'encapsuler le contexte et de permettre sa réutilisation. La généricité du langage UML en fait un langage approprié pour modéliser le contexte. [LL08] modélise le contexte d'une application de gestion du trafic aérien. Mais, le modèle proposé est spécifique à l'application et ne peut être utilisé dans d'autres applications sans modification. Sheng et Benatallah [SB05] ont proposé un méta-modèle basé sur une extension d'UML qui permet de modéliser le contexte auquel des services web sont sensibles au contexte. Ce langage est appelé ContextUML. Comme l'illustre la Figure 2.4, ce méta-modèle est composé de plusieurs classes qui permettent de créer des services sensibles au contexte. La classe Context permet de décrire un contexte observable. Ce dernier peut être un contexte de bas niveau représenté par la classe AtomicContext, ou bien un contexte interprété représenté par

la classe CompositeContext. Pour chaque contexte de bas niveau, le modèle permet de spécifier la source à partir de laquelle il a été collecté.

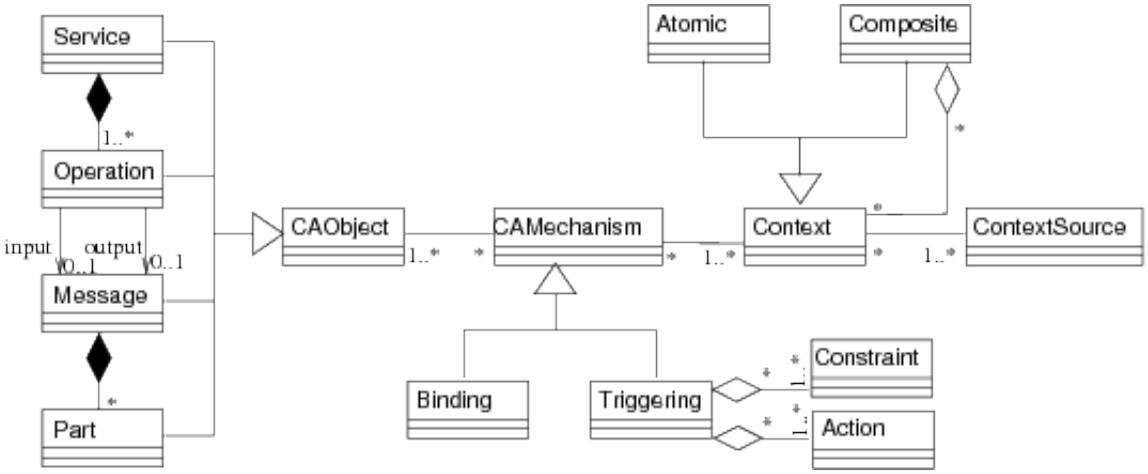

Figure II.4 – ContextUML.

4.4.3 Modèle orienté objet

Utiliser les techniques de l'orienté objet permet de profiter pleinement de la force de ce système (encapsulation, réutilisabilité, héritage etc.). Les approches existantes utilisent des objets variés pour représenter les différents types d'informations contextuelles (localisation etc.) et encapsulent les détails du traitement et de la représentation du contexte dont les accès sont fournis par des interfaces bien définies. Le projet Hydrogen [HSP+03] est basé sur ce modèle.

4.4.4 Modèle de systèmes de balisage

Cette modélisation consiste en un ensemble de balises avec des attributs [BB06]. Dans XML (eXtensible Markup Language), les balises sont définies dans une DTD (Document Type Definition). La description des profils est l'utilisation typique de ce type de modélisation. Pascoe [Pas98] a utilisé un protocole d'échange de données contextuelles au format XML entre un serveur et un utilisateur mobile dans le cadre d'une application appelée Stick-e Note, [Bro96]. Ce protocole baptisé ConteXtML est utilisé d'une part pour décrire le contexte observé associé à l'utilisateur, d'autre part, pour décrire le profil de l'application. Ce profil représente le contexte nécessaire à cette application pour qu'elle puisse s'exécuter.

CC/PP (Composit Capability/Prefrence Profile) est la proposition du W3C pour la représentation de profils [BB06]. C'est un cadriciel basé sur RDF. CC/PP permet de décrire les capacités d'un dispositif ainsi que les préférences de l'utilisateur en utilisant une structure de profils. Cette structure est composée d'une hiérarchie à deux niveaux. Chaque profil a un certain nombre de composants et chaque composant a des caractéristiques, comme l'illustre la Figure 2.5, qui représente le profil d'un client.

Le vocabulaire offert par CC/PP n'est pas riche et a besoin d'être étendu car il est restreint à la description de profil [BB06]. De plus, il ne permet pas la description des relations et des contraintes complexes entre les informations de contexte. [IRRH03] Ont proposé le Comprehensive Structured Context Profile (CSCP) qui permet d'avoir une

```
<rdf:Description rdf:about="http://www.example.com/profile#MyProfile">
  <ccpp:component>
    <rdf:Description rdf:about="http://www.example.com/profile#TerminalHardware">
      <rdf:type rdf:resource="http://www.example.com/schema#HardwarePlatform" />
      <ex:displayWidth>320</ex:displayWidth>
      <ex:displayHeight>200</ex:displayHeight>
    </rdf:Description>
  </ccpp:component>
</rdf:Description>
```

Figure II.5 – CC/PP.

description du contexte non limitée à deux niveaux hiérarchiques. Cette proposition ne permet pas elle non plus de décrire des relations. Les travaux de [HBS02] Ont étendu le vocabulaire de CC/PP pour pouvoir décrire la localisation, les caractéristiques du réseau et les dépendances d'une application.

4.4.5 Modèle à base de grammaire

Stephen et al, [YK01] Ont proposé une grammaire pour décrire un langage de description d'interfaces sensibles au contexte nommé CA-IDL. Ce langage permet d'ajouter la sensibilité au contexte à des applications réparties orientées objet. Le langage CA-IDL permet de décrire les contextes auxquels l'application est sensible, les situations pertinentes et les actions d'adaptation que l'intergiciel doit invoquer lors de leur détection. Comme l'illustre la Figure 2.6, la plate-forme qui fournit le langage CA-IDL, définit trois catégories de contexte observable. Chaque catégorie est constituée d'un nombre limité de types de contexte.

- La catégorie DeviceSpecificContext permet de décrire des informations de contexte spécifiques à une machine ;
- La catégorie EnvironmentSpecificContext permet de décrire des informations de contexte spécifiques à l'environnement qui entoure l'application ;
- La catégorie UserSpecificContext permet de décrire des informations de contexte spécifiques à l'utilisateur.

```
RCSMContext
DeviceSpecificContext {
double battery_power;
double light_intensity;
double net_transmission_rate;
};

RCSMContext
EnvironmentSpecificContext {
unsigned int number_peer_device;
char[16] location;
};

RCSMContext
UserSpecificContext {
unsigned int calendar_usage_rate;
};
```

Figure II.6 – grammaire CA-IDL.

4.4.6 Modèle basé sur la logique

Les modèles basés sur la logique sont caractérisés par un très grand degré de formalité. Ils utilisent l'algèbre booléenne et la logique du premier ordre pour modéliser le contexte. La logique permet de définir des conditions qui nécessitent de déduire des faits ou des expressions à partir d'un autre ensemble d'expressions ou de faits. Par conséquent, dans les modèles basés sur la logique, le contexte est défini comme des faits, des expressions ou des règles. La première approche de modélisation du contexte en utilisant la logique a été publiée en 1993 par McCarthy

et son équipe [McC93]. McCarthy définit le contexte comme une entité mathématique abstraite ayant des propriétés. Ce type de modélisation est utilisé dans le domaine de l'intelligence artificielle, Figure 2.7.

```
[regle1: (?P hasLocation ?x) Equal(?x,'Salle de Reunion')-> Alerte('Salle de Reunion') ]
```

Figure II.7 – modèles basés sur la logique .

4.4.7 Modèle à base d'ontologie

Une ontologie est une description sémantique, structurée et formelle des concepts d'un domaine et de leurs inter-relations [UG96]. Plusieurs modèles d'ontologies ont été proposés pour décrire le contexte. Ces approches permettent non seulement de modéliser le contexte, mais aussi de raisonner sur les données décrites. Les Modèle à base d'ontologies sont caractérisés par une possibilité d'extension et de partage des données. De nombreux langages informatiques sont apparus pour construire et manipuler des ontologies. Dans le but de mettre au point un langage standardisé, le W3C a créé OWL (Ontology Web Language) définissant une syntaxe pour décrire et construire des vocabulaires afin de créer des ontologies, [Hef04]. OWL est un langage basé sur RDF [KC04], il est défini en trois sous-langages de plus en plus expressifs, chacun étant une extension du précédent.

- OWL-Lite est le sous-langage le plus simple. Il est destiné aux utilisateurs qui ont besoin d'une hiérarchie de concepts simples ;
- OWL-DL permet une expressivité bien plus importante. Il est fondé sur la logique descriptive qui est un domaine de recherche étudiant la logique, ce qui confère à OWL-DL son adaptation au raisonnement automatisé ;
- OWL-Full est la version la plus complexe de OWL, mais également celle qui permet le plus haut niveau d'expressivité. Toutefois, les problèmes liés à la complexité de calcul en un temps acceptable le rendent inutilisable.

À chaque ontologie, on peut associer un moteur d'inférence qui permet de raisonner sur les informations de contexte en exécutant des règles d'inférence.

4.5 Synthèse

Les informations du contexte collectées à partir de plusieurs sources peuvent être incohérentes, obsolètes, incorrectes ou incomplètes. La complexité des informations de contexte nécessite des moyens de modélisation pour simplifier leur utilisation. Dans le but d'étudier la possibilité d'utiliser ces approches dans un environnement pervasif, nous évaluons ces approches :

- **L'approches paires/triplets** est caractérisée par une pauvreté d'expressivité et la simplicité des données qu'elles représentent. leur description ne permet pas une description complète du contexte, ni l'expression des relations qui peuvent exister entre les informations de contexte ni l'héritage ni la manière de déduire un contexte de haut niveau De ce fait, elles ne permettent pas de décrire les règles d'interprétation, les actions d'adaptation et les conditions qui permettent de les déclencher. Méthode de récupération : la recherche linéaire.

- **Les approches orientées modèle (Grapique)** sont des approches prometteuses car elles utilisent un modèle formel pour décrire le contexte et offrent aussi un méta-modèle de description qui peut être réutilisé par plusieurs applications ou intergiciels. Les approches orientées modèle existantes ne prennent pas en compte la

description des règles d'interprétation de contextes de haut niveau, ni la modélisation des capteurs. Cependant, ces approches offrent une possibilité d'extension pour permettre la description de nouveaux types de relations entre les informations de contexte et ainsi la possibilité d'intégrer les notions qui manquent. Méthode de récupération : transformation.

- **Modèle basé sur XML** a une structure hiérarchique : Composite Capabilities/Preference Profile (CC/PP), User Agent Profile (UAProf), Méthode de récupération : Markup Query Language.

- **Les approches basées sur la logique** sont des approches formelles dont l'utilisation a été longtemps restreinte au domaine de l'intelligence artificielle. Ces approches permettent de raisonner sur les informations de contexte pour déduire de nouvelles valeurs du contexte ou pour lancer des réactions au niveau de l'application ou du système. Les approches appartenant à cette catégorie permettent de décrire des relations entre les informations de contexte, mais leurs applications aux systèmes existants restent limitées ; cela est dû à la difficulté de description qu'engendre ce type d'approche. Méthode de récupération : inférence.

- **La modélisation à base de gramaire** se distingue par le fait qu'elle permet la description des situations pertinentes ainsi que les méthodes d'adaptation nécessaires si ces situations sont détectées. Mais, elles ne permettent pas de décrire et d'interpréter un contexte de haut niveau. De plus, les modélisations CA-IDL et celle proposée par n'offrent pas la possibilité de décrire les sources de contexte. Méthode de récupération : inférence

- **Les approches à base d'ontologie** sont des approches formelles qui tirent parti des caractéristiques des ontologies pour modéliser le contexte. En effet, les caractéristiques de partage et de distribution des données ont été exploitées afin de définir des méta-modèles de description du contexte. De plus, les moteurs d'inférence fournis par les ontologies ont été utilisés pour déduire des contextes de haut niveau à partir des données collectées. Méthode de récupération : logique premiere ordre.

L'objectif de l'étude que nous avons effectuée consiste à montrer l'apport de chaque approche de modélisation de contexte dans le but d'utiliser l'une d'elles pour modéliser les informations de contexte associées aux applications pervasives. Les exigences mentionnées ci-dessus sont importantes pour toutes les approches de la modélisation de contexte dans les environnements de l'informatique ubiquitaire, cependant l'approche orientée ontologie répond bien à ces critères. En revanche, cela ne veut pas dire que les autres approches sont inutilisables.

L'évolution des systèmes d'information depuis le modèle distribué vers l'informatique ubiquitaire a défini des propriétés désirables des systèmes ubiquitaires et de leurs méthodes de conception vis-à-vis du modèle de contexte citées au chapitre 1, on peut ajouter :

- **La qualité et la richesse de l'information :** est un indicateur de la prise en compte par le modèle de contexte des éventuelles pannes des dispositifs de mesures, erreurs d'interprétation ou de mesure, etc. Un système ubiquitaire met en place des indicateurs de la qualité ou fiabilité de la perception du contexte.
- **Le niveau de formalisme :** dénote l'expressivité atteinte par le modèle dans la description des relations entre plusieurs dimensions du contexte, dans la déduction possible de nouvelles relations et dans la validation formelle de l'ensemble des propositions établissant des liens entre les objets mesures du contexte ;
- **La portabilité aux systèmes existants :** mesure la capacité du modèle de contexte afin être intégré au sein

Tableau II.1 – Propriétés des systèmes ubiquitaires et leurs méthodes de conception vis-à-vis du modèle de contexte, (-),(+) et (0) correspond au degré de prise en charge

	Paire clé/valeur	ontologie	logique	Objet	XML	Graphique UML
Distribution	-	++	++	++	+	-
Hétérogénéité	+	+	o	o	+	-
Proactive	-	++	+	-	-	-
intégration	+	+	-	+	o	-
Qualité information	-	+	+	+	+	+
sécurité	-	++	+	o	o	o
Capacité raisonnement	-	++	+	-	-	-
Scalabilité	-	+	+	o	+	-
invisibilité	+	+	+	+	+	-
interopérabilité	+	+	-	+	++	+
Méta donnes	-	+	-	+	+	+
Degré formalisme	–	++	++	+	+	+
applicabilité	+	+	–	+	++	+
disponibilité	-	+	0	-	+	+
réutilisation	-	+	-	o	o	o
extensibilité	-	+	++	++	+	-

de systèmes d'information (mobiles, distribues ou autres) initialement insensibles au contexte.

Les exigences mentionnées ci-dessus sont importantes pour toutes les approches de la modélisation de contexte dans les environnements de l'informatique ubiquitaire ont des exigences, cependant l'approche à base d'ontologie répond bien à ces critères est la plus prometteuse en termes de raisonnement et d'expressivité, Table 2.1.

4.6 Analyse du contexte

L'analyse du contexte est un processus qui contrôle continuellement les données collectées [BB06]. Ce contrôle permet de filtrer les observations et les interprétations du contexte pour détecter les situations pertinentes dans le but d'adapter le système en conséquence ou de notifier les applications sensibles à ces situations pertinentes. Le développement du processus d'analyse nécessite une interaction avec le processus de collecte du contexte via le dépositaire de contexte pour vérifier les valeurs du contexte et détecter leurs changements pertinents. Grâce aux résultats obtenus par cette analyse, des décisions d'adaptation sont prises par le processus d'adaptation afin d'adapter le comportement de l'application aux nouvelles situations pertinentes.

4.7 Couche de dissémination du contexte

Cette couche assure la transmission des différentes informations contextuelles à l'application [Cha07]. Ces informations sont distribuées sur différents lieux géographiques et proviennent de plusieurs types de dispositifs. Ceci a poussé les chercheurs à dédier une couche à la transmission des données a l'application. Cette couche assure une transparence totale de la communication avec l'application. En conséquence, le développement de l'application devient plus simple. Sans cette couche, on serait amené à développer des protocoles de communications avec les différentes sources de contexte. La couche de dissémination du contexte offre des moyens de communication standards pour notifier l'application des changements de contextes et leur transmission à l'application. Plusieurs systèmes offrent des

mécanismes de gestion d'évènements qui se basent essentiellement sur les fonctions de gestion de requêtes directes ou de notification. L'application peut demander un accès direct à une information contextuelle précise (pull) mais elle peut aussi s'abonner pour recevoir tous les changements des valeurs de cette information (publish/subscribe). Ces deux fonctions principales assurent un moyen de communication transparent et efficace pour la dissémination des valeurs de contexte à l'application. L'implantation de ces types de communication est nécessaire pour garantir la sensibilité au contexte dans une application.

4.8 Couche application

La couche application dans les systèmes sensibles au contexte existants est représentée par l'application qui offre ses services aux différents clients concernés [Cha07]. Cette couche est responsable de l'extraction des informations des différentes sources de données attachées à l'application. Elle doit aussi implémenter les réactions nécessaires aux changements du contexte. Chaque application s'abonne à la couche de dissémination du contexte pour accéder aux différentes informations contextuelles et être informée de leurs changements. Une application peut accéder à ces informations de deux façons différentes : synchrone et asynchrone. Dans le cas d'un accès synchrone, l'application demande à la couche de dissémination de lui fournir une information contextuelle précise. Ceci est réalisé généralement par des appels directs de fonctions au niveau de la couche dissémination. Dans le cas de la communication asynchrone, l'application s'abonne à des évènements spécifiques qui correspondent à des changements de valeurs de contexte. Dès qu'un événement est déclenché, l'application est simplement notifiée ou bien l'un de ses services est directement invoqué en utilisant des fonctions de callback implémentées dans la couche dissémination (par exemple, la technique de callback dans CORBA.

5 Principales plateformes existantes de sensibilite au contexte dans les systemes d'informations pervasifs.

Le développement et l'exécution des applications sensibles au contexte peuvent être effectués à l'aide des intergiciels sensibles au contexte [BB06] en intégrant dans leurs architectures des entités qui se chargent de gérer le contexte et l'adaptation de l'application. Des premiers prototypes dont le deploiment est resté confiné aux laboratoires, datant des années 90, les premiers travaux en ubiqiuté proposent principalement des services liés à la localisation, Nous nous intéressons spécifiquement aux prototypes suivants :

5.1 ActiveBadge

Le projet ActiveBadge [WHFG92] developpe par la firme Olivetti a permis la realisation d'une application permettant l'acheminement des appels telephoniques à une personne selon sa localisation au telephone le plus proche de d'elle. Dans ce type d'organisation, souvent un réceptionniste est charger de transmettre les appels téléphoniques vers les postes internes, dans les bureaux des employés. Il dispose habituellement d'un plan avec les codes des téléphones associés aux utilisateurs. Expérimenté en 1990 au laboratoire de recherche Olivetti en Angleterre, ce projet est généralement considéré comme le premier exemple d'application sensible au contexte. Le projet vise à une meilleure localisation et coordination d'un large groupe de personnes dans une entreprise. Le système utilise des badges émettant des signaux infrarouges (à une fréquence bien déterminée). Ces badges sont portés par le personnel d'une entreprise et chaque badge contient l'identification du porteur, Figure 2.8.

Figure II.8 – ActiveBadge.

5.2 ParcTab

Le projet ParcTab de Xerox, [WSA+95] est une infrastructure qui favorise le développement des applications sensibles au contexte de localisation (localisation d'une personne, matériel, etc.). Le ParcTab est un assistant personnel digital (PDA) porté par l'usager et fonctionne comme un terminal graphique. Il est en communication infrarouge avec des émetteurs-récepteurs, Figure 2.9. Ces émetteurs-récepteurs communiquent à leur tour (en infrarouge) avec des passerelles. Ces passerelles sont connectées à un réseau local de postes de travail au moyen d'une connexion RS-232. À chaque ParcTab correspond un agent logiciel qui contrôle la communication avec des applications qui s'exécutent sur les postes de travail du réseau local. Cela décharge les ParcTabs des opérations de traitement qui épuisent leurs ressources limitées. Le système de communication est basé sur le mécanisme d'appel de procédure à distance (RPC) entre les ParcTabs et les applications sur les postes de travail du réseau local. Il transmis la notification des courries pour un utilisateur selon sa localisation et les personnes avoisinants par un texte qui s'affiche par le ParcTab ou une simple notification par un signal sonore (le cas échéant de filtrer les courriels urgents lorsque l'utilisateur se trouve en conférence ou réunion).

Figure II.9 – ParcTab.

5.3 Stick-e-notes

Le projet stick-e-notes, [Pas98] consiste en un cadre de travail pour le développement des applications sensibles au contexte principalement de localisation. Il est basé sur l'utilisation d'un ensemble de PDAs connectés à un capteur de localisation (GPS ou ActiveBadge). Ces PDAs peuvent être en communication ou non selon l'application. L'idée de base est inspirée des bouts de papiers collant (stick notes) qu'on met généralement sur les portes, tables, équipements, etc. Pour se rappeler de quelque chose ou attirer l'attention de quelqu'un. Ces notes seront électroniques et écrites par l'utilisateur et attachées à des contextes bien précis (exemple : localisation) et sauvegardées dans son PDA, Figure

2.10. Ces notes sont déclenchées (affichées ou signalées) plus tard lorsque le même contexte apparaît de nouveau (exemple : l'utilisateur attache une note descriptive d'un musée lorsqu'il se trouve dans ce dernier. Chaque fois que l'utilisateur entre dans cette zone géographique, la note descriptive du musée est affichée). Les notes peuvent être de différents formats : texte, pages HTML, son, vidéo, programme à exécuter.

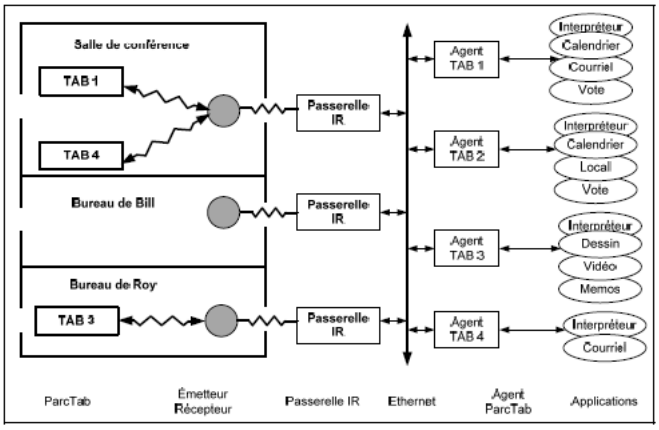

Figure II.10 – Stick-e-notes.

5.4 CyberGuide

L'idée du projet cyberguide, [AAH+97] est d'équiper l'utilisateur d'un guide touristique électronique sensible à son contexte (localisation, orientation). L'infrastructure matérielle est composée d'un ensemble d'assistants personnels digitaux (PDAs) connectés à des GPS pour détecter la position du touriste. Ces assistants personnels peuvent communiquer entre eux ou avec un réseau local par infrarouge, Figure 2.11. L'objectif est de guider un touriste dans sa visite touristique en lui offrant les sites intéressants à visiter selon la localisation et les chemins à suivre ainsi que des informations utiles selon sa position.

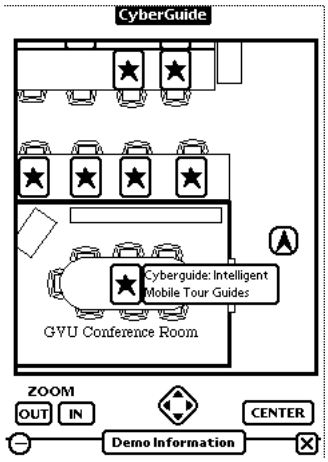

Figure II.11 – Cyberguide.

5.5 Classroom 2000

Dans le domaine de l'education, le projet eClass, [AAF+96] (connu initialement sous le nom Classroom 2000) est une salle de classe augmentée mise en place à l'universite, Figure 2.12. Son role est de capturer les interactions presentes lors d'un cours. La salle possede un systeme d'enregistrement audio et video, un tableau blanc electronique ainsi que des systemes de prises de notes individuels. L'objectif de cet environnement est de capturer les echanges d'information qui ont lieu lors d'un cours et de permettre d'y acceder ulterieurement facilement a l'aide d'une interface Web. On peut egalement citer le projet Smart Classroom ([XSXX01] ; [SXX+03]) de l'Universite de Tsinghua. Ce projet propose egalement d'augmenter une salle de classe traditionnelle (video-projecteurs, tableau tactile, microphones et cameras) afin de permettre une integration naturelle du tele-enseignement au sein d'une classe traditionnelle.

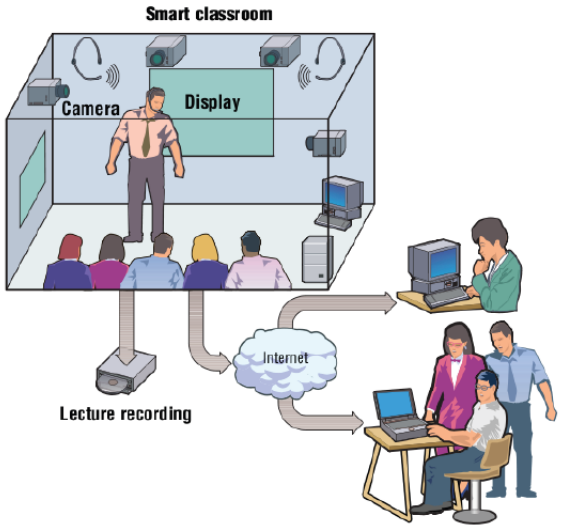

Figure II.12 – Classroom 2000.

Les architectures proposées sont la plupart spécifiques à un domaine d'application (systèmes de localisationet demandent des efforts supplémentaires pour l'analyse et l'adaptation. Les architectures fondées sur un serveur de contexte souffrent du problème principal d'un système centralisé : lorsque le serveur tombe en panne, tous les autres composants seront affectés. Cela contredit la nature de l'information contextuelle dans un environnement pervasif qui est en général distribuée.

5.6 Tableau Récapitulatif

Le Tableau 2.2 résume les caractéristiques des architectures étudiées :

Tableau II.2 – Caractéristiques des architectures sensibles au contexte étudiées

	Type Architecture	Modèle	capture	interprétation
ActiveBadge	Client/Serveur	Attribut-Valeur	Emetteur infrarouge	–
ParcTab	Client/Serveur	Attribut-Valeur	IR	–
Cyberguide	Hybride	–	GPS, IR	–
Stick-e-notes	Client/Serveur	–	GPS	–
Classroom 2000	Distribué	–	Capteurs audio et video	–

6 Principales plateformes de gestion du contexte dans les systemes d'informations pervasifs

6.1 Introduction

Les informations issues de l'environnement physique et matériel créent un contexte pour l'interaction entre les humains et les services informatiques. Le Contexte est constitué de toute information qui caractérise une situation en relation avec les interactions entre les utilisations, les applications et l'environnement physique. Pour rendre les applications sensibles au contexte, il est nécessaire de fournir des outils permettant de le capturer, le comprendre et le traiter pour pouvoir influer sur le comportement des applications. Trois champs d'action sont définis :

- le support informatique ;
- les périphériques et le réseau ;
- l'utilisateur et l'environnement physique.

6.2 Le Context Toolkit

Le Context Toolkit [DAS01] fait partie des premiers travaux dans le domaine de la sensibilité du contexte dans les systèmes pervasifs [Cha07]. C'est un Framework Java générique qui permet le prototypage rapide des applications sensibles au contexte. Le context toolkit est inspiré des boites à outils des interfaces graphiques (Java Context Toolkit), il fait la médiation entre l'application et l'information de contexte capturée. Il utilise pour cela un ensemble de composants abstraits widgets, interpréteurs et agrégateur communiquant entre eux par TCP/IP et XML. Ils permettent l'acquisition des informations de contexte à partir des capteurs et leur transformation en information de contexte de haut niveau.

Les travaux autour de Context Toolkit de Dey [DAS01], ont permis de catégoriser les composants de calcul de contexte en trois types : les widgets, les interprètes et les agrégateurs.

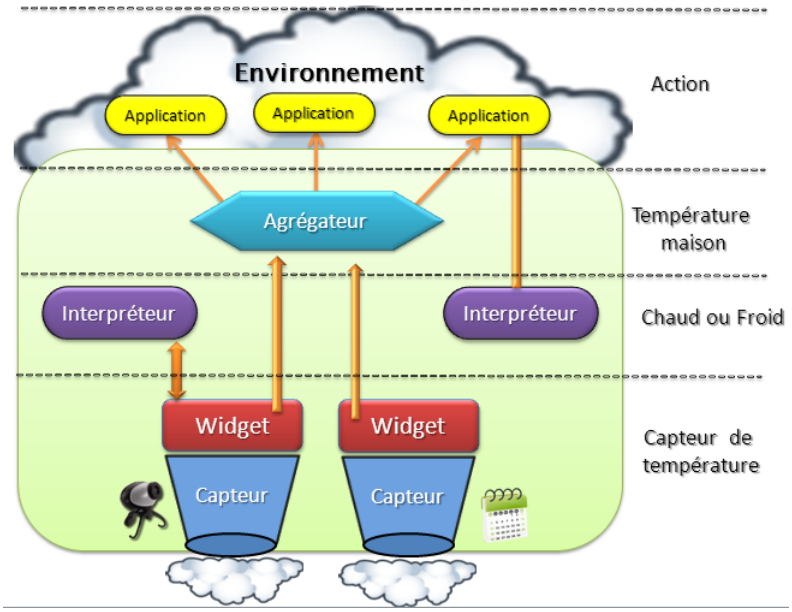

Figure II.13 – Architecture du Context Toolkit.

- Les widgets : sont des composants logiciels autonomes qui encapsulent les capteurs physiques ou logiciels qui capturent l'information de contexte de bas niveau (par exemple un widget capteur de température). À travers ces composants, les applications peuvent accéder aux données sur un certain élément de contexte sans, pour autant, manipuler directement les technologies utilisées pour sa détection. Son rôle aussi est de communiquer à un (ou plusieurs) serveur(s) ou interpréteur(s) les informations perçues en fonction des données reçues par le capteur. Il a aussi pour rôle de constituer un historique qui lui est propre.
- Les interprètes : Ils permettent de relever le niveau d'abstraction d'une information de contexte. Ils prennent une donnée bas-niveau capturée par un ou plusieurs widgets et la traduisent vers une donnée de plus haut niveau. Par exemple : un interprète indique s'il fait chaud ou froid en fonction d'une température, Figure 2.13,
  - un interpréteur a la capacité de convertir des coordonnées GPS en adresse (n, rue, ville).
  - un interpréteur de localisation de personne peut fournir comme informations :
    Personne A dans la salle B20.
    Personne A dans le Bâtiment B de l'institut.
    Personne A dans le Campus de Sidi Bel Abbes.
- les agrégateurs : rassemblent les informations de contexte données de widgets en une seule donnée : entité (une personne, un objet, un endroit...), agissant ainsi comme une passerelle entre le contexte et le système. Ils agrègent toutes les informations collectées et interprétées concernant une entité donnée. (par exemple un agrégateur donne la température d'une maison en regroupant les données de tous les capteurs de température de la maison). Les connexions entre les context widgets, les agrégateurs et les applications ont lieu par l'intermédiaire du discoverer.
- les services : sont un type particulier de widget qui, au lieu de collecter les informations, contrôlent ou modifient ces informations dans l'environnement. Ces composants peuvent être découverts dynamiquement par les applications à travers un autre composant, le " discoverer ", auprès duquel les composants s'enregistrent à leur démarrage.
- le Discoverer : est chargé de maintenir à jour l'annuaire des composants disponibles et de décrire leurs propriétés comme le langage, le protocole de communication, le nom de la machine sur laquelle ils sont disponibles, etc. 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 :

1. En première analyse, le modèle est clair et introduit la notion de niveau d'abstraction.
2. Le context toolkit n'offre pas de mécanismes permettant de raisonner sur le contexte tels qu'un moteur d'inférence à cause du modèle utilisé pour le représenter (clé/valeur).
3. Le context toolkit n'offre pas non plus de mécanismes d'adaptation qui peuvent être utilisés par des applications pour réagir aux différents changements de contexte d'un utilisateur mobile.
4. Un modèle ne fournit pas de métadonnées pour renseigner l'application sur la qualité et/ou la précision des informations transmises à l'application qui doit donc faire confiance ou pratiquer ces calculs elle-même.

5. Le mécanisme de prise en charge des événements (pour notifier le changement du contexte) consiste à utiliser un processus léger (thread) pour chaque événement, ce qui implique une charge importante pour le système.
6. Bien que les composants du Context Toolkit puissent être répartis, le Discoverer doit être connu de tous. Ce point central constitue un point faible par rapport tant au passage à mobilité (le Discoverer ne sera pas toujours accessible pour tous les clients mobiles).
7. la sécurité et la protection de la sphère privée ne sont pas traitées.

6.3 GAIA

Est un intergiciel [RHC+02] développé par l'université de l'Illinois pour la création d'infrastructures active spaces, c'est à dire d'environnements ubiquitaires sensibles au contexte centrées sur l'utilisateur [RC02]. Ranganathan et Campel [RC02] 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. Gaia ajoute à la prise en compte du contexte, l'adaptabilité en fonction de la localisation. Ce système d'exploitation pour applications d'informatique diffusé 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.

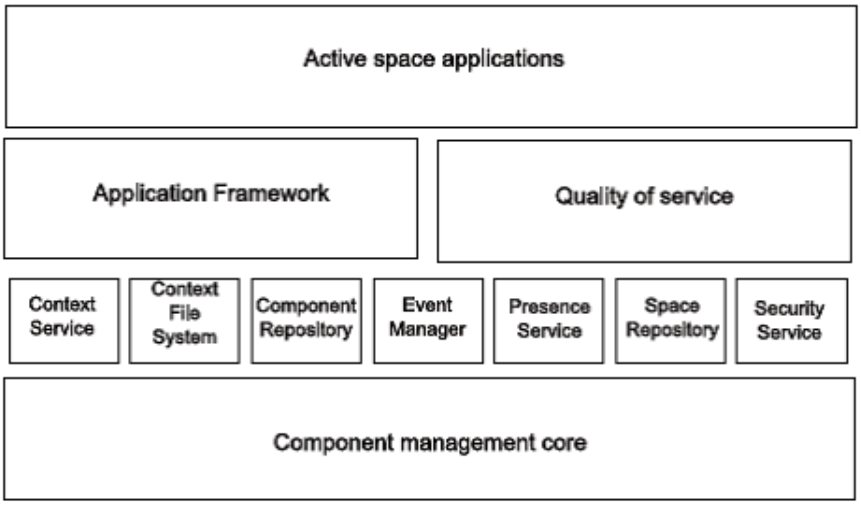

Figure II.14 – GAIA.

L'infrastructures active spaces est administrée par une entité logicielle sensible au contexte permettant la gestion de dispositifs hétérogènes. Les applications sur Gaia sont à base de composants et distribuées, la gestion de ces composants pouvant se faire de manière distante. La forte dynamicité de l'environnement, et par conséquent la variabilité du contexte, ont impliqué l'utilisation des événements afin d'être réactif à ces variations. Cette tâche est accomplie grâce à l'event manager. Ce middleware étant resources-aware, il est nécessaire de posséder des informations relatives aux entités présentes dans un active space (que ce soit des applications, utilisateurs, dispositifs ou encore des services). Ainsi le physical entity presence détecte les nouvelles entités dans l'environnement ubiquitaire.

Le contexte est pris en compte grâce au context service qui se compose de context providers. Ceux-ci fournissent aux applications des informations relatives au contexte, obtenues par différents capteurs. D'autre part ce service

comprend également des composants capables de déduire des informations de plus haut niveau à partir de celles obtenues par les providers. Enfin un annuaire indique les providers disponibles, Figure 2.14.

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. Le contexte est représenté avec des prédicats de logique du première ordre. Le nom du prédicat représente, par convention, le type de contexte (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, aussi des opérations de quantification, ainsi que l'opérateur de déduction qui peut servir à écrire des règles.

Analyse :

1. Gaia souhaite adapter pour les environnements d'intelligence ambiante, cette infrastructure offre un éventail riche de services transversaux.
2. 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.
3. Le modèle de contexte proposé est interopérable et basé sur la logique du premier ordre.
4. L'utilisation des ontologies permet une compréhension uniforme des prédicats entre les différents agents. Cependant, il faut que les agent se réfèrent à une ontologie particulière. Si on souhaite faire interopérer plusieurs environnements, il doit exister des correspondances entre les ontologies étrangères et l'ontologie de référence de l'environnement.

6.4 Context Broker Architecture(CoBrA)

CoBrA (Context Broker Architecture) [CFJ03] est une architecture centralisée basée sur un agent courtier (broker) d'une entité physique ou virtuelle pour appuyer le développement des applications sensibles au contexte. Par exemple, salles de réunion intelligentes, maisons intelligentes, et véhicules intelligents. L'architecture contient les éléments suivants, Figure 2.15 :

- L'agent Context broker : est l'élément central de l'architecture, il permet de gérer le modéle de contexte pour les différents agents et périphériques qui y sont connectés. Chaque agent permet d'obtenir des informations du contexte à partir des capteurs et les transformer en un modèle de contexte formel afin qu'ils puissent être partagés par tous les agents. L'agent de contexte est composé de quatre modules principaux :
  1. La base de connaissance du contexte : contient le modèle de contexte de chaque domaine. L'ontologie du domaine est un ensemble de concepts et de relations entre eux dans un domaine spécifié.
  2. Le module de raisonnement du contexte : est responsable du raisonnement sur un domaine donné à travers des règles logiques d'inférence. Le module de raisonnement agit par rapport aux contraintes sur les relations entre les concepts dans l'ontologie du domaine. Le module de maintien de la connaissance vérifie l' uniformité des informations de la base de connaissance du contexte.
  3. Le module d'acquisition de contexte : est le middleware qui permet de récupérer les informations de contexte de bas niveau à partir des divers capteurs. Ce module a une architecture en deux couches : les capteurs du contexte et l'interprétateur du contexte. Les capteurs physiques ou virtuels, fournissent les informations du contexte. L'interprétateur du contexte récupère ces informations et les interprète.

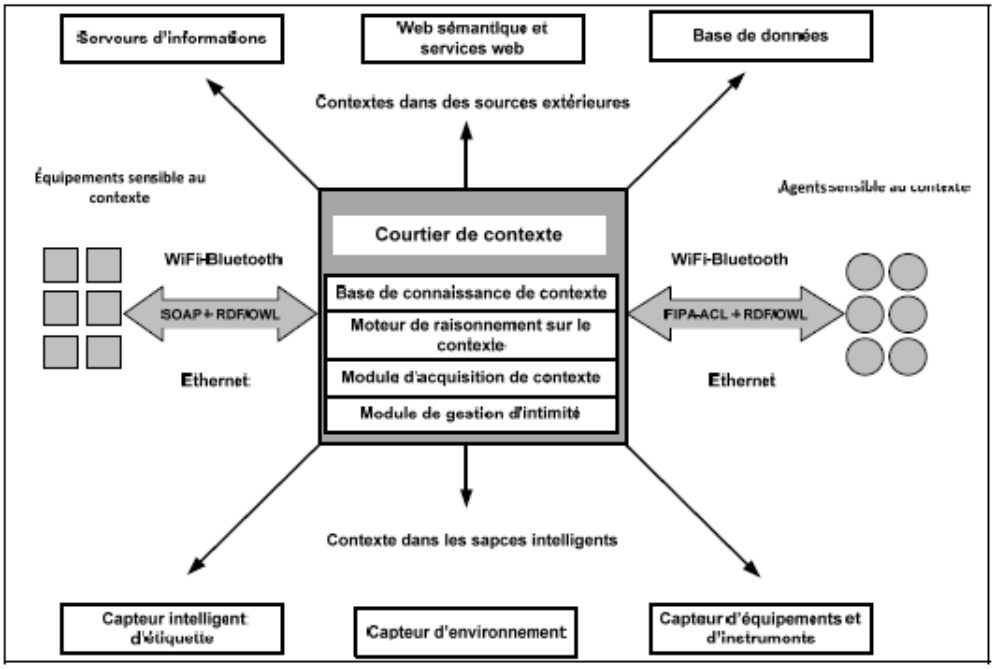

Figure II.15 – Architecture de COBRA.

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

Dans l'application choisie par l'auteur [CFJ03] : salles de réunions intelligentes , les domaines représentent des parties du monde réel. La base de connaissances est couplée avec un moteur d'inférence composé de trois parties : le module de raisonnement sur l'ontologie déduit des faits à partir des informations de la base de connaissances et des modèles définis dans les ontologies ; le module de raisonnement sur le contexte est un moteur d'inférence hybride basé sur la logique (déduction, logique flux, etc.) et l'analyse statistique (arbre de décision, réseau Bayesien, etc.) ; et le module de maintien de la cohérence des connaissances sert à supprimer le bruit dans les informations de contexte en provenance de l'environnement physique et à réconcilier la base de connaissances avec les changements trop rapides de l'environnement.

Pour le modèle de contexte, CoBrA se base sur l'approche à base d'ontologie, Figure 2.16 . CoBrA-Ont est un ensemble d'ontologies exprimées dans le langage OWL et qui définissent les concepts et les relations qui les lient dans chaque domaine. Cet ensemble contient 41 classes et 36 propriétés qui sont catégorisées selon des domaines distincts : places, agents, contexte de localisation des agents et contexte d'activité des agents. Elle utilise un modèle centralisé pour le stockage et le traitement du contexte puisque les équipements mobiles dans un système pervasif possèdent des ressources limitées et une politique de confidentialité pour l'utilisateur.

Chaque ovale de ce graphe représente une classe dans le langage OWL. Les ovales avec des lignes entrecoupées indiquent le type de l'information que le Context Broker recevra des autres agents et de capteurs de l'environnement. Les éléments de contexte sur lesquels est basé la gestion du contexte sont "Person ", " Place " et " Intention " ;

– La classe Person : définit les propriétés générales d'une personne dans un espace intelligent. Dans CoBrA, chaque personne doit avoir un nom, une adresse email et une homepage URL. Parmi les sous classes qui dérivent de la classe " Person " :

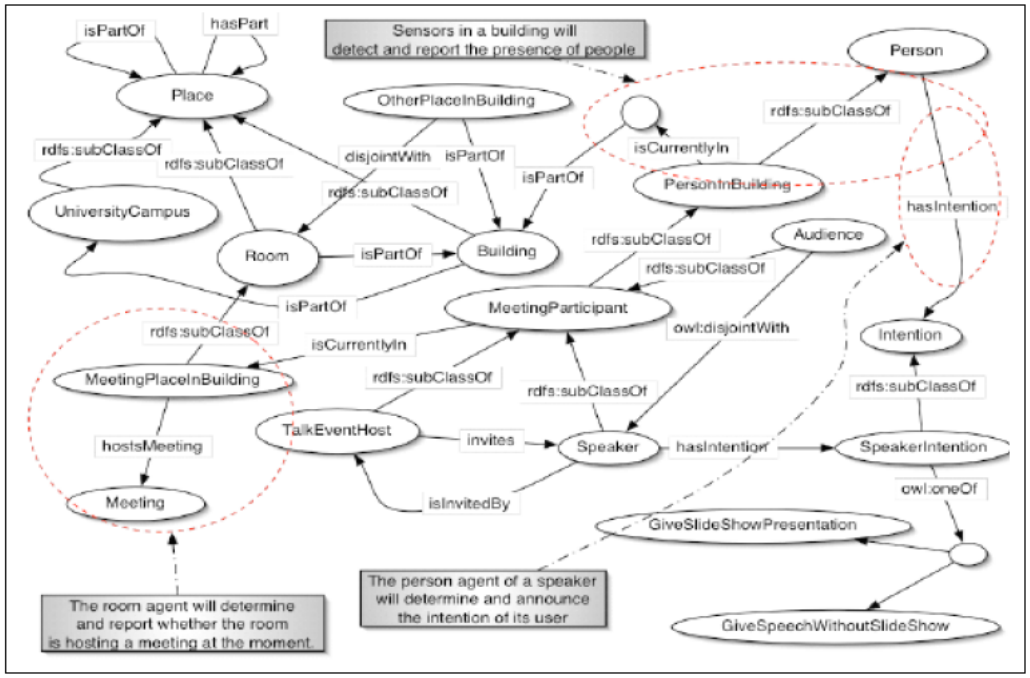

Figure II.16 – Ontologie de COBRA.

- La classe PersonInBuilding : définit une des personnes qui sont actuellement dans un bâtiment.
- La classe MeetingParticipant : définit un type de personnes qui sont dans une réunion.
- La classe Place : définit les propriétés indiquant l'appartenance à un tel endroit ; isPartOf et hasPartOf. Les sous-classes décrites dans cette ontologie sont : "UniversityCampus", "Building", "Room", "OtherPlaceInBuilding" et "MeetingPlaceInBuilding",
- La classe intention : indique la notion des intentions de l'utilisateur. Par exemple l'intention d'un interlocuteur de donner une présentation et l'intention d'une assistance de recevoir une copie de diapositives de la représentation. Cette classe possède une seule sous classe, "SpeakerIntention" L'adaptation au contexte est réalisée à partir des propriétés et des descriptions des classes et les relations entre ces classes que possède le langage OWL.

Analyse :

1. l'architecture proposée fait une séparation du processus de capture de contexte et son utilisation. Cela permet une abstraction des détails de capture de bas niveau et un accroissement de l'extensibilité et de la réutilisation du système.
2. CoBrA offre cependant un haut niveau de confidentialité et de sécurité pour l'échange des informations grâce au module de politique de confidentialité des agents.
3. L'architecture proposée est la plupart spécifiques à un domaine d'application (systèmes de localisation, interaction personne-machine, etc.) et demande des efforts supplémentaires pour l'adaptation.
4. L'architecture fondée sur un serveur de contexte souffre du problème principal d'un système centralisé : lorsque le serveur tombe en panne, tous les autres composants seront affectés. Cela contredit la nature de l'information contextuelle dans un environnement diffus qui est en général distribuée.
5. L'architecture nécessite un serveur dédié pour l'agent courtier, ce qui surcharge le système et augmente les

risques de congestion au niveau du serveur.

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

6.5 Context Management Framework(CMF)

Le context management framework (CMF) [KMK+03] est une plateforme analogue au context toolkit Dey [DAS01]. Elle est composée de quatre entités principales, Figure 2.17 : resource server (capture de contexte), context recognition service (interprétation du contexte), context manager (dissémination du contexte) et application [Cha07]. Comme la plateforme CoBrA [CFJ03], le context management framework se base sur un gestionnaire de contexte centralisé qui communique avec tous les autres modules de la plateforme. En effet, dans CMF, le gestionnaire de contexte récupère les informations contextuelles à l'aide du module resource server. Ensuite, il les interprète en utilisant le module context recognition service. Enfin, il les diffuse à l'application. Cette dernière peut communiquer avec le context manager (gestionnaire de contexte) de trois manières différentes :

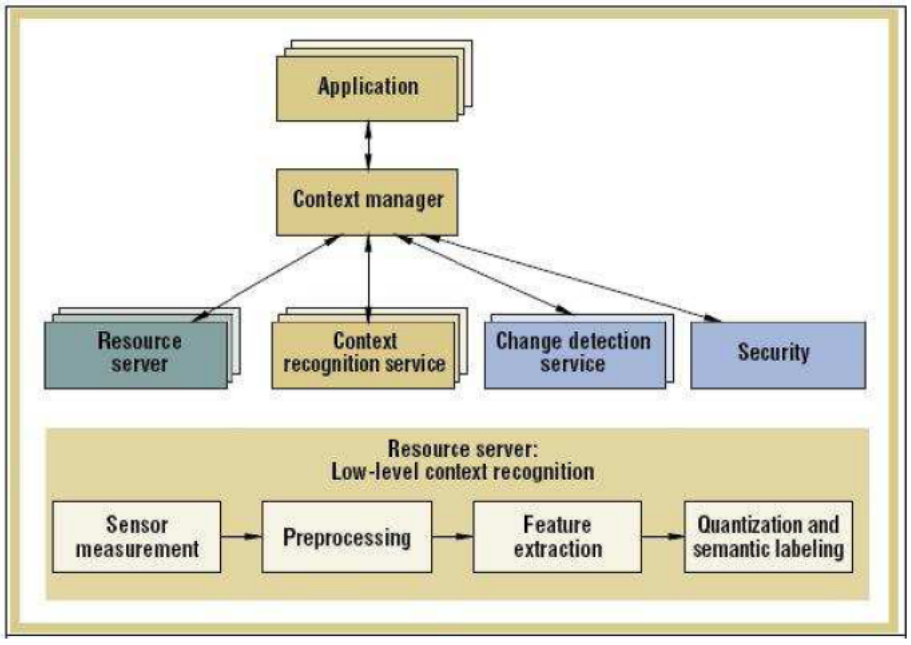

Figure II.17 – Architecture de CMF.

- Elle envoie des requêtes directes au context manager. Dans ce cas, le context manager délègue cette requête au resource server ;
- Elle s'abonne aux changements des valeurs de contexte. Dans ce cas, le context manager utilise le service de détection des changements de contexte " change detection service " ;
- Elle demande un contexte de haut niveau (composé) en utilisant le service d'interprétation du contexte context recognition service.

Analyse :

1. Dans CMF, nous remarquons que le stockage du contexte n'est pas centralisé. Ceci constitue une différence majeure par rapport aux autres plateformes présentées.
2. Une autre caractéristique intéressante de CMF, consiste à l'utilisation de la logique floue pour fournir des informations contextuelles complexes. Les autres plateformes assument que le contexte (même de haut niveau) est bien spécifié et représenté alors que CMF considère que l'obtention du contexte de haut niveau n'est ni simple ni directe. Les auteurs de CMF [KMK+03] proposent donc d'utiliser la logique floue pour obtenir des informations de haut niveau et ainsi enrichir le contexte et la description de l'environnement de l'utilisateur.

6.6 CASS

CASS (Context-Awareness Sub-Structures)[FC04] est un intergiciel pour la sensibilité au contexte qui permet aux développeurs de gérer les applications sensibles au contexte. Cette gestion concerne l'interaction avec les capteurs pour collecter les informations de contexte, la sauvegarde des données collectées dans une base de données et leur interprétation. . Il offre une bonne abstraction des informations contextuelles et utilise un modèle orienté objet UML pour la représentation du contexte où les modules sont représentés sous forme de classes, Figure 2.18. L'écouteur des capteurs reçoit les mises à jour de la part des noeuds de capteurs et enregistre les informations du contexte reçues dans la base de données. L'écouteur des capteurs est relié à l'écouteur des changements qui est responsable de communiquer aux périphériques le changement de contexte. Le récupérateur du contexte récupère les données stockées dans la base de données.

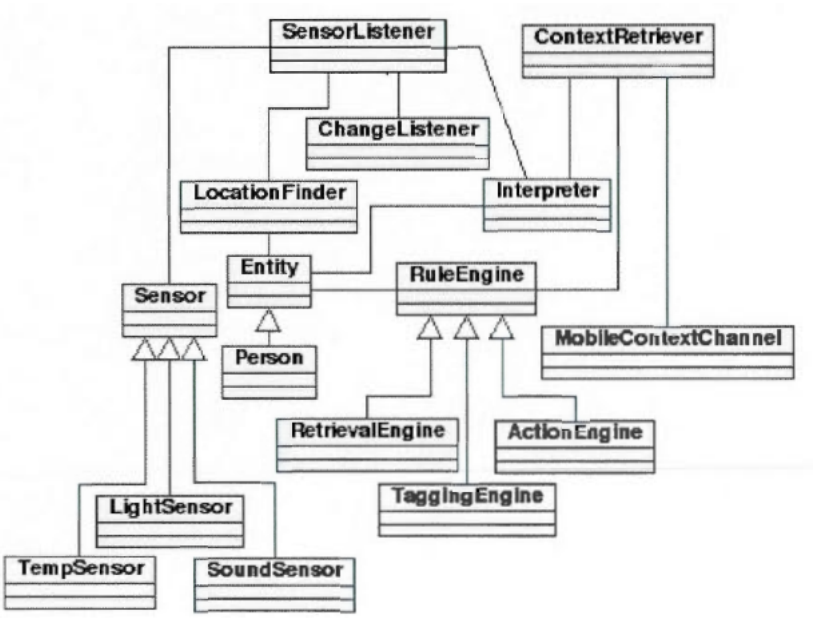

Figure II.18 – CASS.

l'intergiciel CASS est structuré en un ensemble de composants pour gérer le contexte, Figure 2.19.

– **SensorLister** se charge d'interagir avec des noeuds de capteurs pour collecter le contexte et de le sauvegarder dans la base de données.

– **Le composant Interpreter** se charge de calculer des informations de contexte de haut niveau en utilisant le RuleEngine et **le ContextRetriever** se charge de retrouver les observations du contexte dans la base de données.

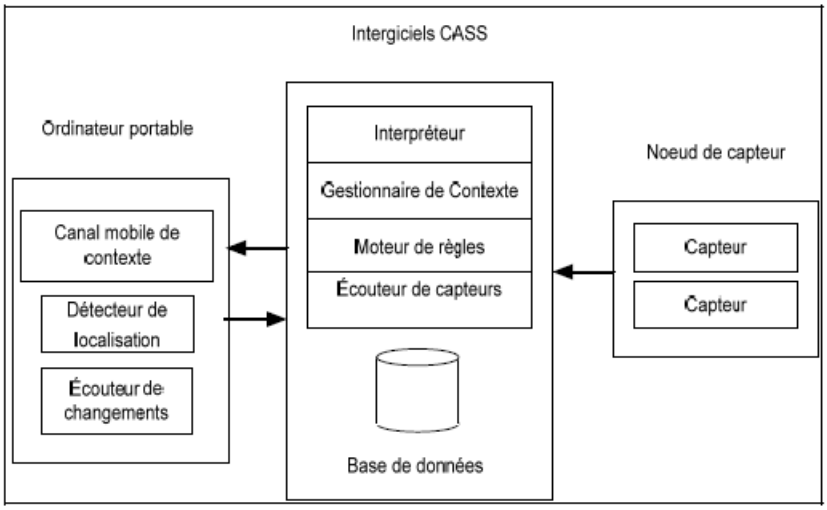

Figure II.19 – Architecture CASS.

L'intergiciel se charge de détecter le changemement du contexte observé et de notifier les applications sensibles à ces contextes. Chaque application se charge d'analyser ce nouveau contexte et d'appliquer l'adaptation nécessaire.Cette architecture est basée sur un serveur contenant une base de données des informations contextuelles ainsi qu'une base de connaissances. Celle-ci permet au moteur d'inférence d'inférer d'autres informations de contexte en utilisant la technique de chaînage arrière. La base de connaissances contient les règles de production. Elle est stockée sous formes de tables dans la base de données relationnelles. Les équipements mobiles sont munis de différents types de capteurs. Ceux-ci perçoivent les changements du contexte et les envoient au serveur sans aucun traitement local. La communication entre les équipements mobiles et le serveur est assurée par des connexions sans fils. Le serveur contient aussi un module pour l'interprétation du contexte perçu. Cette architecture offre une bonne modularité permettant à un utilisateur de modifier facilement les composants du serveur, en particulier le mécanisme d'inférence.
Analyse :

1. L'intergiciel CASS facilite l'interaction avec les capteurs et l'interprétation des données en intégrant dans son architecture des composants qui se chargent d'effectuer ces tâches. Les applications sensibles au contexte doivent s'enregistrer auprès de cet intergiciel pour être notifiées à chaque fois qu'une variation de contexte a été détectée. Le développeur de l'application se charge de programmer l'analyse de ces données ainsi que l'adaptation de l'application. Ce dernier ne peut pas choisir les capteurs que l'intergiciel doit utiliser pour collecter les informations de contexte.
2. Les équipements mobiles dans CASS ne font aucun traitement local (tout se fait au niveau du serveur). Cela limite leurs autonomies, en revanche elle favorise l'extensibilité du système.
3. CASS offre une bonne interprétation du contexte, donc plus d'abstraction. Le module de raisonnement qui lui permet d'être plus proactif.
4. La structure centralisée quand à elle présente une faiblesse (en cas de panne du serveur).

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

6.7 CORTEX

Biegel et Cahill, [SBBC04] ont proposé un Framework pour faciliter le développement des applications sensibles au contexte qui s'appelle CORTEX (Cooperating Real-time sentTient objects). L'implémentation de cet intergiciel est basée sur OpenCOM [CBCP01], qui est lui meme un intergiciel réflexif basé sur la technologie COM [Box97]. Ce qui a l'avantage de rendre la plateforme efficace, légère et réflexive. L'architecture est basée sur les " sentient objects " qui ont des caractéristiques avantageuses pour un environnement informatique diffusé. Ces caractéristiques sont les suivantes :

- Elles ont la capacité de percevoir l'état de l'environnement par des capteurs ;
- Elles sont autonomes en ayant la capacité de fonctionner indépendamment du contrôle humain d'une manière décentralisée ;
- Elles sont proactives en prenant des initiatives pour accomplir un but proposé.

L'architecture de ce middleware est composée suivant 4 structures de composants, Figure 2.20 [DIBSS04] :

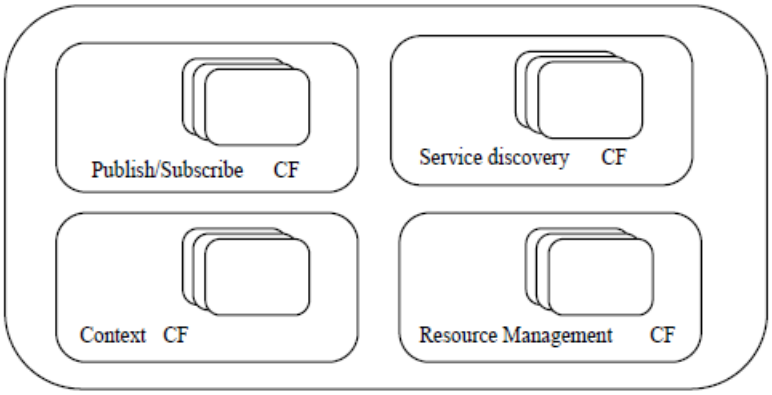

Figure II.20 – Architecture de Cortex.

- Publish/subscribe : fonctionne suivant un modèle d'événements. Avec le composant " publisher " qui a pour rôle d'envoyer un événement alors que le composant " subscriber " a pour rôle de le recevoir.
- Service discovery : permet la découverte de service qui a été annoncé par les protocoles de découvertes de service comme SLP, UPnP.
- Context : permet de gérer les informations du contexte par le stockage des informations provenant d'objets sensibles lors de la phase de consommation d'événement et en sens inverse de déclencher des événements.
- Ressource management : permet de gérer les ressources.

Le cadre de composants Context est basé sur le modèle des objets sensibles, Figure 2.21. Un objet sensible est composé de trois entités qui fournissent respectivement des fonctions de collecte de contexte (Sensory Capture), de représentation de contexte (Context Hierarchy) et de raisonnement sur les informations de contexte (Inference Engine). La fonction de collecte de contexte interagit avec des capteurs à travers des évènements. La fonction de représentation de contexte permet de modéliser le contexte d'une manière hiérarchique en décrivant les attributs de

chaque contexte. Dans le cadre de CORTEX, les attributs les plus importants d'un contexte sont : les situations pertinentes auxquelles le composant est sensible et l'ensemble des règles actives dans ces situations. La fonction de raisonnement contient des règles qui dictent les réactions de l'objet sensible dans des situations particulières, ces règles sont décrites à l'aide du langage CLIPS.

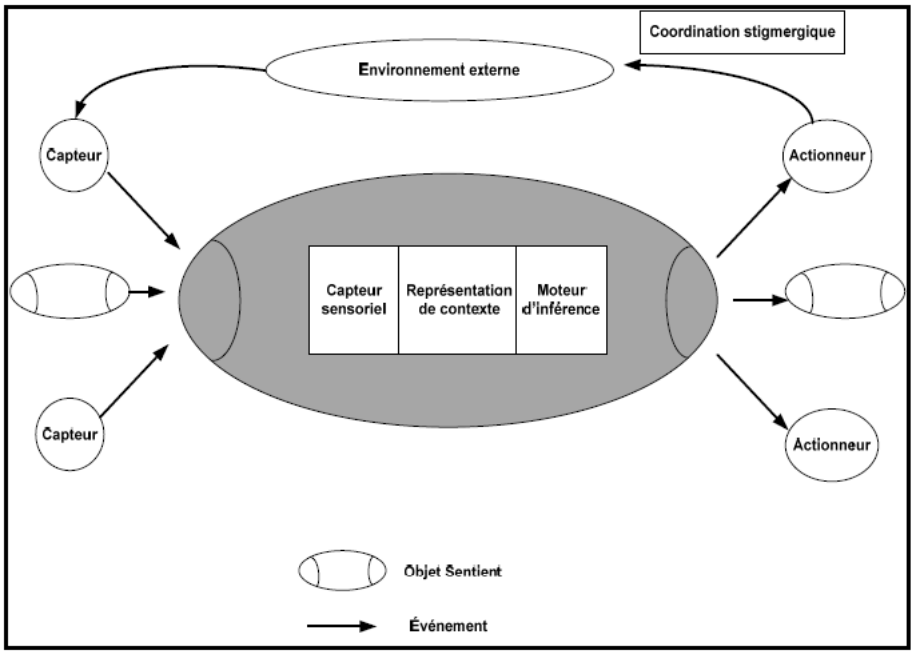

Figure II.21 – Modèle d'objets sensibles de Cortex.

Analyse :

1. CORTEX facilite la création d'applications sensibles au contexte en fournissant une interface graphique qui aident le développeur dans cette tâche. L'interface graphique fournit un ensemble de sensors et d'actuators prédéfinis qui peuvent être utilisés par les développeurs d'applications.
2. L'intergiciel CORTEX offre aux développeurs d'applications un modèle hiérarchique pour décrire le contexte et le moyen de choisir les capteurs avec lesquels l'application doit interagir. Il se charge aussi d'adapter le comportement de l'application aux situations pertinentes du contexte à l'aide du moteur d'inférence.
3. l'interprétation du contexte de haut niveau n'est pas considérée par cet intergiciel.
4. Le système d'inférence écrit en CLIPS demande des développeurs qualifiés pour le mettre en œuvre dans une autre application, ce qui limite son utilisation.
5. CORTEX est utilisé avec des applications à petite échelle.
6. Le fait que CORTEX utilise un modèle à événement le rend très approprié pour un environnement ad hoc dans lequel les déconnexions sont imprévisibles.

6.8 Service oriented context-aware middleware(SOCAM)

SOCAM est un intergiciel proposé par l'université de Singapour qui vise à assurer le développement et le prototypage rapide de services sensibles au contexte dans des environnements intelligents [GPZ05]. Ce projet propose un

middleware distribué orienté service qui convertit les divers espaces physiques en un espace sémantique où le contexte peut être partagé et fourni à des services sensibles au contexte. Le middleware offre la possibilité d'intégrer et de réutiliser aisément des composantes dans le système telles que des nouveaux capteurs [GPZ05]. Ce système consiste en des composantes organisées en modules qui communiquent et interagissent entre eux et permettent d'acquérir les informations du contexte à partir de différentes sources distribuées. Il permettent aussi de traiter et d'inférer le contexte en se basant sur un module de raisonnement et de livrer l'information du contexte aux services du contexte localisés en haut de l'architecture.

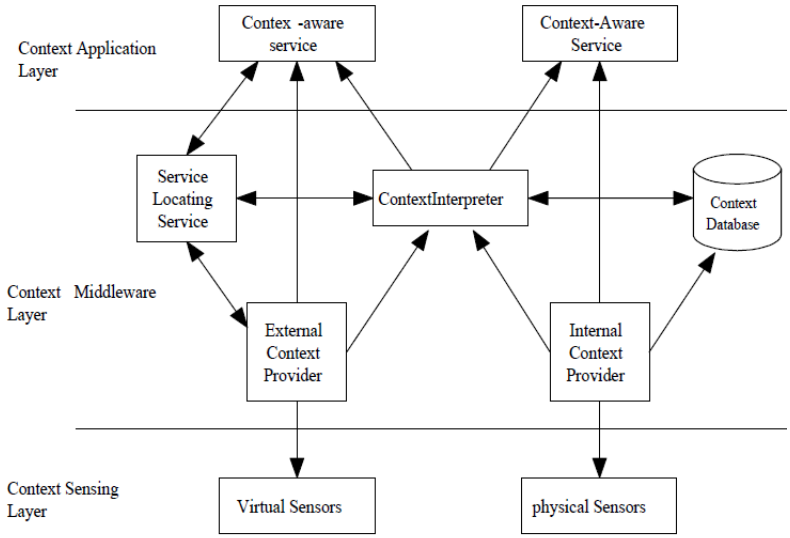

Figure II.22 – SOCAM.

L'architecture de SOCAM utilise le modèle client/serveur, elle suit un modèle en couche, Figure 2.22 :

- La couche de captage : est constituée des différents capteurs utilisés pour saisir les données du contexte ils peuvent être physiques comme un GPS ou un détecteur de mouvement, ou virtuels comme un service Web ;
- Context Providers (fournisseurs de contexte) : Les fournisseurs du contexte récupèrent les données de différentes sources de contexte constituées des capteurs physiques et des capteurs virtuels. Ensuite, ils les convertissent en des représentations OWL pour que le contexte puisse être partagé et réutilisé par d'autres composants de l'architecture. Il existe deux types de Context provider :
  - **Les Context providers externes** se chargent de collecter le contexte à partir de sources de données externes à la machine comme la température de la pièce par exemple ;
  - **Les Context providers internes** se chargent de collecter des données sur l'état interne de la machine comme la valeur de la bande passante par exemple.
- Context interpreter (interprète de contexte) : C'est un module qui permet de traiter le contexte à l'aide d'un raisonnement logique. La fonction de ce module est d'inférer le contexte de haut niveau à partir des informations de contexte de bas niveau comme la géolocalisation. Ce module est composé d'un raisonneur. Ce dernier est constitué des règles d'inférence qui sont prédéfinies par les experts du domaine en forme de logique de premier ordre ;
- Context database (base de données de contexte) : Elle stocke les différents éléments de l'ontologie pervasive

décrivant l'environnement de l'application et les instances des ontologies spécifiques au domaine de l'application décrivant l'environnement de l'utilisateur ;

- Context-aware services (services sensibles au contexte) : Ces services utilisent les différentes informations stockées dans la base de données de contexte (contexte bas niveau et/ou contexte interprété) pour modifier leur comportement selon le contexte courant. Pour cela, chaque service s'enregistre auprès du Context provider pour acquérir le contexte auquel il est sensible (contexte pertinent) et fournit au Context interpreter des règles d'interprétation qui permettent à l'intergiciel d'interpréter le contexte ;
- Service-locating service (service de localisation de services) : Service est équivalent aux pages blanches, Il est utilisé par le Context Interpreter et le Context Provider pour s'enregistrer. Il fournit un mécanisme avec lequel des utilisateurs ou des applications peuvent localiser les fournisseurs et les interpréteurs de contexte pour un domaine donné.

Pour le modèle du contexte, les auteurs adoptent l'approche basée sur les ontologies (CONON) qui permetent une description sémantique du contexte, Figure 2.23. Le langage utilisé pour la description des ontologies est OWL. La conception des ontologies est faite d'une manière hiérarchique sur deux niveaux qui réduisent la taille et la complexité de l'ontologie. L'ontologie de haut niveau définit les concepts généraux tels que : personne, localisation, activité et entité computationnelle [Cha07]. L'ontologie spécifique au domaine est un ensemble d'ontologies de bas niveau qui représentent les concepts généraux dans chaque sous-domaine et leurs Propriétés. Deux ontologies de domaine spécifique ont été développées : maison avec 89 classes et 156 propriétés et véhicule avec 32 classes et 57 propriétés. L'ontologie de bas niveau doit être mise à jour dynamiquement lors des changements de contexte.

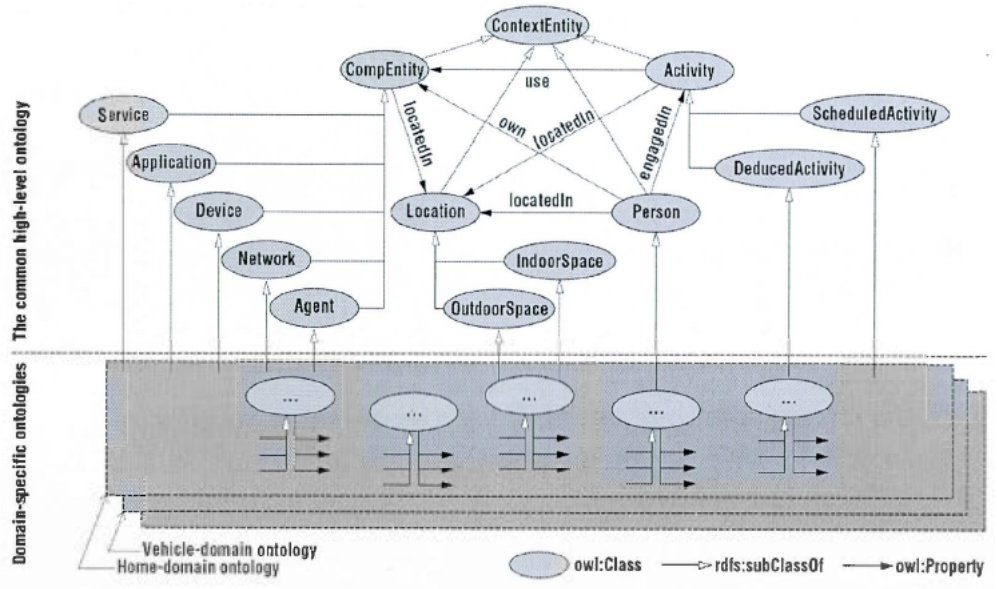

Figure II.23 – L'ontologie SOCAM.

Analyse :

1. L'architecture est utilisée pour le développement d'une petite application (voiture intelligente). Évidement cela limite son utilisation dans des domaines variés de l'informatique pervasive.
2. L'interpréteur du contexte est chargé par une quantité importante sous forme d'ontologies de différents domaines. Ce choix affecte la performance globale du système et présente les inconvénients de l'architecture

centralisée, mais favorise la réutilisation.

3. L'intergiciel SOCAM décharge les développeurs de services sensibles au contexte de la tâche de collecte et d'interprétation des données, il leur fournit un modèle pour décrire le contexte à observer.
4. les développeurs doivent implémenter l'analyse et l'adaptation de chaque service.
5. Cet intergiciel n'offre aucun moyen aux développeurs pour choisir les capteurs que l'intergiciel doit utiliser afin de collecter le contexte.

6.9 Smart-M3

C'est une infrastructure pour les espaces intelligents de code source libre de Nokia [Rod11]. Cette architecture repose sur le modèle blackboard et un modèle de contexte ontologique. L'élément central de cette architecture est la composante de courtage sémantique (Sematic Information Broker ou SIB). Les processeurs de connaissance (Knowledge Processor ou KP), quant à eux, accèdent et traitent l'information.

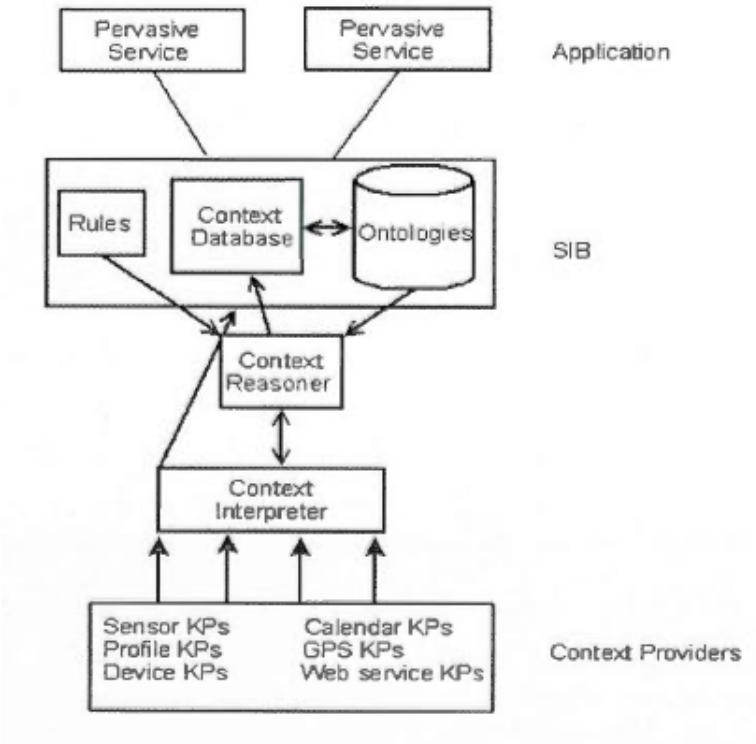

Figure II.24 – Architecture Smart-M3.

L'architecture, Figure2.24 est composée des éléments suivants :

– Fournisseurs de contexte (context provider) : donnent les informations atomiques concernant le contexte de l' utilisateur qui peuvent être des données de bas niveau ou des données du service Web.
– Interpréteur du type des donnés de contexte : les données brutes issues de diverses sources hétérogènes doivent être converties en données sémantiques qui peuvent être utilisées par les différents services.
– Raisonneur et interpréteur des règles du contexte : cette partie infére de nouvelles données de haut niveau à partir des informations atomiques du contexte. Ces règles sont définies par les développeurs de processeurs

d'information.

- Ontologies : représentent les données issues des fournisseurs de contexte et celle inférées par le raisonneur.
- Règles d'inférence : sont nécessaires au processus d'inférence du raisonneur. Ces règles se basent sur la logique et définissent la manière d'inférer l'activité de l'utilisateur à partir des informations du contexte données par le fournisseur du contexte (KP).

Le modèle de contexte Figure 2.25 de cette infrastructure est basé sur les ontologies. Le contexte de l'utilisateur peut être divisé en deux parties : contexte atomique et contexte inféré. Le contexte atomique est représenté par les données du contexte obtenues directement des fournisseurs d'informations du contexte (date, localisation, appareil). Le contexte inféré est composé des informations déduites des données de contexte (activité, occupation, association). Le modèle ontologique est divisé en deux paties : les ontologies de haut niveau et les ontologies du domaine. Le contexte de l'utilisateur est représenté par six concepts : temps, emplacement, périphérique, activité, occupation et association.

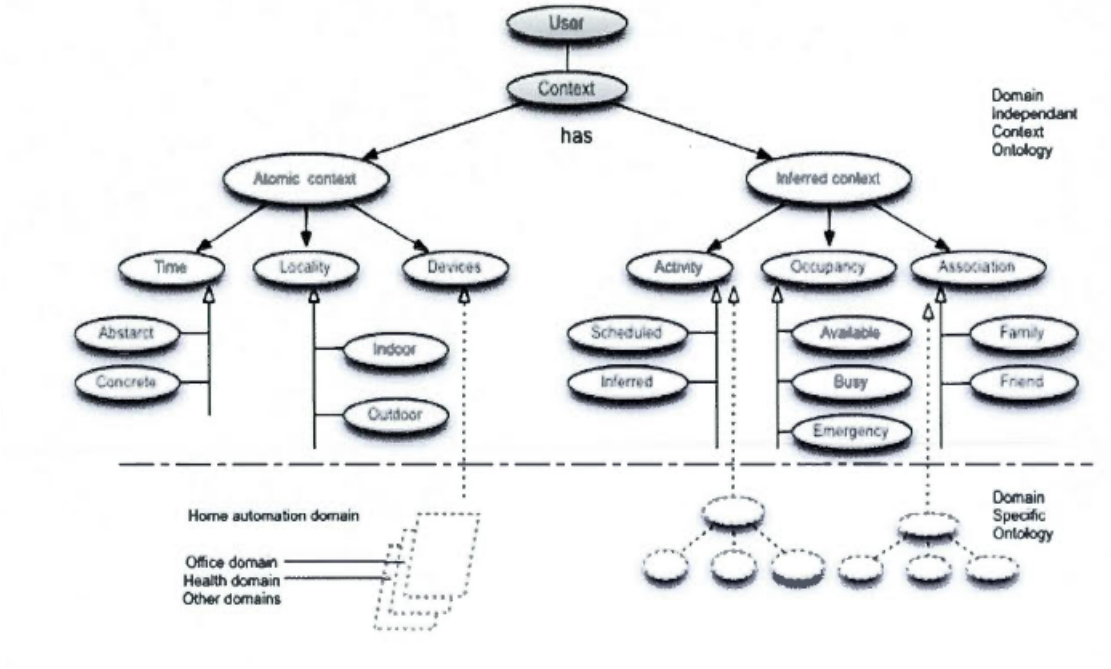

Figure II.25 – Ontologie Smart-M3.

Analyse :

1. L'architecture Smart-M3 en s'appuyant sur le modèle blackboard, flexible aux changements dynamiques de l'environnement de l'utilisateur.
2. Le modèle de contexte permet de supporter les sources d'information hétérogènes et offre l'extensibilité et flexibilité du système ainsi qu'une forte expressivité.

6.10 Synthèse

Le tableau 2.3 synthetise les differentes approches etudiées et met en lumiere leurs points de convergence et de divergence.

- Le Context Toolkit fait partie des premiers travaux des applications sensibles au contexte. Il propose un canevas logiciel pour aider le concepteur de l'application a modeliser le contexte en vue d'une utilisation facilitée. Il propose des elements de base comme la capture, l'interpretation et l'agregation.
- Les plateformes réutilisent le modèle général à base de cinq couches présenté dans la Figure 2.3.
- Dans ces plateformes, il y a un accord commun sur la séparation de l'acquisition du contexte d'une part et de son utilisation d'autre part. En effet, il est très important d'encapsuler les mécanismes d'acquisition du contexte dans des composants qui offrent une interface de communication standard.
- Il n'y a pas un modèle standard d'acquisition de contexte à partir de plusieurs sources différentes. Chaque plateforme propose son propre modèle d'acquisition et de collecte de contexte. SOCAM utilise des capteurs encapsulés par des services web et la communication avec l'interpréteur du contexte est réalisée par un échange d'événements.
- les plateformes existantes se distinguent aussi par le modèle de contexte utilisé.
- L'utilisation des ontologies offre une grande richesse sémantique et plus de possibilité d'inférences et d'interprétation du contexte par rapport aux autres modèles de contexte. Cependant, leur implémentation reste difficile à réaliser surtout dans les cas d'utilisations de plusieurs ontologies comme dans le cas de SOCAM.
- L'utilisation d'un modèle trop simpliste comme les couples (attribut, valeur) est source de conflits d'interprétation et de description du contexte actuel de l'application et de l'utilisateur.
- La gestion d'historique du contexte est aussi un autre critère important dans les systèmes sensibles au contexte. En effet, l'historique permet d'implémenter des algorithmes d'apprentissage pour fournir des services hautement adaptables au contexte. De plus, avec ce genre d'algorithmes, des actions proactives peuvent être automatiquement déclenchées pour un certain nombre de services à l'utilisateur sans qu'il formule une demande explicite.
- Un autre aspect important dans ces systèmes concerne la gestion de la sécurité et de la confidentialité des données. En effet, des concepts doivent être spécifiés pour définir à qui appartient l'information contextuelle. CoBrA utilise le langage Rei pour définir les politiques de sécurité en termes de droits et d'autorisations d'accès au contexte. Le context Toolkit implémente le concept d'appartenance du contexte à un utilisateur ou à une entité. Ainsi, une information contextuelle n'est accessible qu'à l'utilisateur ou l'entité lauquelle appartient.
- Tous les travaux cités se basent sur le fait que la réaction de l'application par rapport aux changements du contexte capturé doit être assurée d'une manière ad hoc et spécifique par l'application elle-même. Ces travaux considèrent que les modifications du comportement de l'application restent une tâche propre au développeur de l'application et qu'elles restent un paramètre à prendre en compte lors de son développement.

Tableau II.3 – Tableau synthetisant les differentes approches etudiées

	Architecture	Modèle Contexte	capture	Analyse	Historique Stockage	Découverte Ressources	Confidence
Context ToolKit	Widgets Orienté objet	Attribut-Valeur	Widget de context	Transformation et agrégation	Disponible	Disponible dans un serveur	Appartenant au contexte
COBRA	Centralisé Basé sur agent	Onthologie (OWL) SOUPA	Module aquisition	Moteur inférence	Disponible	Non Disponible	le langage Re
C.M.F	Centrer gestionnaire de contexte	Onthologie (RDF)	Serveur de ressource	Contexte recognition service	Non Disponible	Mecanisme de subscription	Non Disponible
SOCAM	Middleware distribué + serveur centraliisé	Onthologie (OWL)	Fournisseur de contexte	Moteur inférence	Basede donné	Localisation service	Non Disponible
CASS	Centralisé Objet	Modèle de donnée relationnel	Sensor nodes	Moteur inférence	Non Disponible	Disponible uPnP	Non Disponible
GAIA	MVC (Etendu)	Ary Predicates (DAML+OIL)	Context provider	First ordre logic	Service Discovery	Non Disponible	supported
Smart-M3	Centrer gestionnaire de contexte	Onthologie	fournisseurs d'informations du contexte	basent sur la logique	Non Disponible	Non Disponible	Non Disponible
CORTEX	Sentient Object (Component) Event	Modèle de donnée relationnel	capteurs	système d'inférence CLIPS	Non Disponible	Disponible	Non Disponible

7 Conclusion

Au début de ce chapitre, nous avons donné une définition du terme contexte puis nous avons montré l'importance de sa prise en compte dans le domaine de l'informatique pervasive. Les ressources limitées de ces dispositifs et les caractéristiques hétérogènes et dynamiques des systèmes engendrent la nécessité d'utiliser un nouveau type d'applications qui détectent le changement de l'environnement qui les entoure et qui s'adaptent en conséquence.

Les contextes pertinents utilisés par ces applications sont collectés à partir de sources hétérogènes ; ils sont soit explicitement fournis par l'utilisateur, soit capturés à partir de sondes ou encore dérivés (interprétés) à partir d'autres informations de contexte. De ce fait, ils peuvent être sujets à des ambiguïtés, des incohérences ou des imperfections. Les caractéristiques variées des informations de contexte nécessitent des modèles de description abstraits. Nous avons décrit plusieurs approches de modélisation. Après avoir analysé ces approches, nous avons conclu que les approches à base d'ontologies sont les mieux adaptées pour décrire le contexte dans les systèmes ubiquitaires.

La modélisation du contexte est la première démarche dans le processus de création d'applications sensibles au contexte. Elle permet de simplifier le développement de ce type d'applications, mais cela reste insuffisant, car la création de ce type d'applications nécessite plusieurs étapes de programmations supplémentaires, entre autres l'interaction de l'application avec les capteurs pour collecter, analyser ces données, et le lancement des adaptations nécessaires quand une situation pertinente est détectée. Pour cacher à l'application tous ces détails, il faut utiliser un intergiciel qui se charge d'automatiser toutes ces tâches. Dans le chapitre 3, nous étudions l'apport de l'approche service dans les systemes d'informations pervasifs.

Chapitre III

Systèmes d'Information Pervasifs et l'orientation service

1 Introduction

Le paradigme de l'orientation service est présenté comme réponse aux problèmes intergiciels posés par l'ouverture des environnements pervasifs. Les objectifs du travail présenté sont de simplifier voir de masquer trois des préoccupations principales des applications pervasives : les aspects distribués de la découverte et de la communication entre entités pervasives.

la notion de service a été largement étudiée dans la littérature. Elle peut être vue, d'une manière générale, comme étant des composants logiciels qui sont désignés de telle sorte qu'ils supportent des demandes métiers évolutives. Cette notion de service n'a cessé d'évoluer avec l'apparition des services Web et du Web sémantique permettant de garantir un meilleur niveau d'interopérabilité, d'intégration, de compréhension et d'automatisation. Cette évolution a encouragé d'autres chercheurs à étudier la notion de services sous un angle plus proche de l'utilisateur et de son espace. Ceci a fait apparaître les services sensibles au contexte, qui doivent s'adapter aux changements de l'environnement.

Dans le cadre des Systèmes d'Information Pervasifs (SIP), l'orientation service permet de répondre au besoin de gestion de l'interoperabilité et de l'hétérogénéité technique de l'environnement dans lequel évoluent ces systèmes et des actions que ces systèmes proposent afin de satisfaire les besoins des utilisateurs. En effet, l'indépendance des services par rapport aux aspects technologiques et à leurs implémentations est un des moyens permettant de masquer l'hétérogénéité technologique des environnements pervasifs. Avec l'évolution des SI vers les SIP, de nouveaux challenges sont apparus ou ont pris une forme différente. En effet, afin de construire un SIP qui prend en considération le caractère dynamique des environnements pervasifs, il est important de répondre à certains défis, tels que l'adaptabilité, la découverte, la composition. Dans le cadre de cette thèse, nous soulevons la problématique de la selection et l'invocation de services de façon proactif.

Ce chapitre présente un état de l'art sur l'orientation service. Nous commençons par présenter la notion de services et l'émergence des visions techniques. L'orientation service est un concept clé dans la conception des SIP. Nous présentons, les différents challanges et défis auxquels un SIP orienté service doit répondre.

2 La notion de service

A travers la littérature, la notion de service correspond à un concept largement répandu. Originairement, la notion de service est apparue avec l'émergence de l'informatique orientée services (SOC) (Service Oriented Computing) afin de gérer le problème d'interopérabilité entre des applications et des architectures hétérogènes [PG03]. Le paradigme SOC utilise les services comme étant les éléments fondamentaux afin de supporter un développement rapide, fiable et à moindre coût d'applications logicielles distribuées dans des environnements hétérogènes avec une facilité de composition [PTDL07].

La notion de service est employée afin de fournir des abstractions de haut niveau pour l'organisation d'applications à grande échelle sur des environnements ouverts. Cette notion est matérialisée par un composant logiciel souvent associé à un ensemble de fonctionnalités particulières dont l'interface est clairement définie et standarisé, et dont le fonctionnement interne est inconnu des clients. Les services exécutent des fonctions qui peuvent être aussi bien de simples requêtes dans un formulaire, que des processus métiers complexes. Ces services sont conçus comme un ensemble de modules logiciels autonomes qui peuvent être exposés, publiés, découverts, composés et négociés à la demande d'un client et invoqués par d'autres applications [PTDL07]. Le service se comporte ainsi comme une boîte noire : rien de son contenu interne n'est visible aux clients, seule son interface l'est. La notion de service joue un rôle clé dans l'architecture SOA et devient l'unité informatique de base pour supporter le développement et la composition de services de plus en plus complexes, qui à leur tour peuvent être utilisés pour créer des applications flexibles et dynamiques.

Un autre aspect clé se dégage de ces multiples définitions : le faible couplage entre le client et le fournisseur de service. En effet, l'absence de dépendance entre les fournisseurs de services et leurs consommateurs permet d'ordonner les services dans de nombreux flux [FWK02]. Le client n'a pas besoin de connaître la manière dont le service fonctionne ou est implémenté pour faire appel à ses fonctionnalités. C'est ce faible couplage qui rend la notion de service particulièrement attractive pour les environnements pervasifs, puisque ces environnements se caractérisent par la volatilité de leurs éléments. La notion de service se comporte ainsi comme une boîte noire rien de son contenu interne n'est visible aux clients, seule son interface l'est.

3 L'architecture orientée services : SOA

L'architecture SOA est un concept de modélisation métier et une architecture technique présentée sous la forme d'une infrastructure standardisée basée sur XML et sur des Services Web [KGD+06]. Comme, illustre la Figure 3.1, un fournisseur de services peut publier une interface bien définie sur un répertoire qui permet à d'autres parties prenantes de le récupérer et de coupler faiblement ce service offert à leurs propres services. Selon cette définition, les caractéristiques principales d'une architecture SOA sont le faible couplage, l'indépendance par rapport aux aspects technologiques et l'extensibilité. Le faible couplage permet de garantir la réutilisation et l'interopérabilité des services. L'indépendance aux technologies est garantie grâce aux contrats d'utilisation qui sont indépendants de la plateforme technique utilisée par les fournisseurs de services. Enfin, l'extensibilité est rendue possible par le fait que de nouveaux services peuvent être découverts et invoqués à l'exécution.

L'architecture SOA fournit la capacité d'adresser les exigences de l'informatique distribuée, à savoir : indépendance des protocoles, faible couplage, réutilisation et standardisation [PH07]. L'architecture SOA est ainsi mise en place afin de garantir, d'une part, la flexibilité dans la maintenance et l'évolution des systèmes, et d'assurer, d'autre part, un niveau élevé d'interopérabilité entres des systèmes hétérogènes et d'adaptation aux changements.

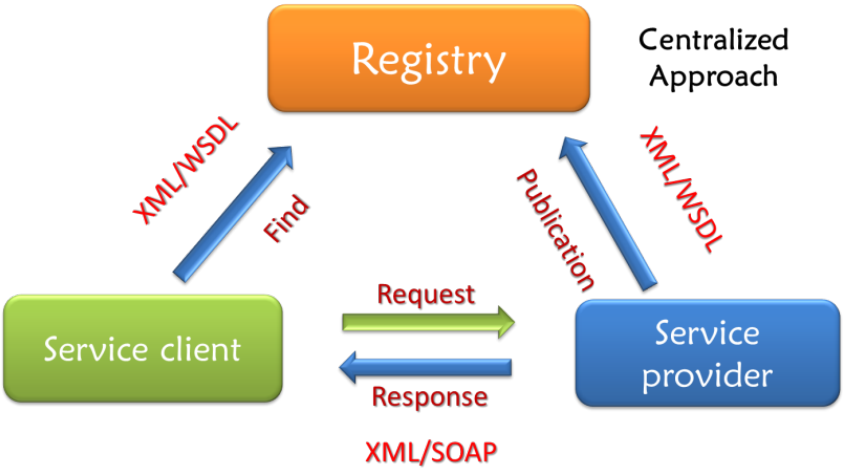

Figure III.1 – L'architecture orientée services : SOA.

L'architecture SOA est utilisée afin de permettre d'une part l'identification et la publication de services pour les rendre réutilisables par les autres systèmes et d'autre part la découverte et la localisation de services [Org06]. Les trois principaux acteurs qui interviennent dans cette architecture : Le fournisseur de services se charge de décrire les fonctionnalités offertes par un service ainsi que les informations nécessaires pour permettre à un client de l'utiliser. Ceci représente la description de services qui sera par la suite publiée par le fournisseur dans un répertoire de services (ou annuaire de services). Ce dernier joue le rôle d'intermédiaire entre le fournisseur et le consommateur de services. Ce consommateur peut interagir avec le répertoire de services afin de retrouver le service qui répond à ses besoins. Cette étape correspond à la phase de découverte de services, laquelle se base sur les descriptions publiées afin de ne sélectionner que celles qui pouvent satisfaire le client (consommateur de service). Si le client trouve le service désiré, alors il peut mettre en place un contrat avec le fournisseur de services, afin de consommer le service.

Le principe de conception derrière l'architecture SOA est qu'un service est une unité faiblement couplée composée d'une interface et d'une implémentation. L'architecture SOA se caractérise ainsi par la séparation entre la description de service (interface) et sa mise en œuvre (implémentation). L'interface définit l'identité d'un service, ses moyens d'invocation et ses capacités fonctionnelles, tandis que l'implémentation représente la mise en œuvre des opérations internes que le service doit exécuter. Cette séparation a permis aux fournisseurs et aux consommateurs de services d'être faiblement couplés. En outre, les services peuvent être facilement réutilisés. Parce que les interfaces de services sont indépendantes de la plateforme et que la mise en oeuvre est transparente pour les consommateurs de services, un client devrait être capable d'utiliser le service à partir de n'importe quel terminal de communication en utilisant n'importe quelle plateforme informatique, système d'exploitation et n'importe quel langage de programmation. Ainsi, l'architecture SOA introduit essentiellement une nouvelle philosophie pour le développement d'applications distribuées, où les services peuvent être découverts, composés, publiés, réutilisés, et invoqués au niveau de l'interface, indépendamment de la technologie spécifique utilisée en interne pour implémenter chaque service [GDG10].

4 Standards des services Web

4.1 HTTP

(Hyper Text Transfer Protocol) est un protocole de connexion de donnees entre les explorateurs web et les serveurs. Il est le standard actuel de transfert des documents HTML. Il est concu pour etre extensible a d'autre format tel que XML. Le but du protocole HTTP est de permettre un transfert de fichiers (essentiellement au format HTML) localises grace a une chaine de caracteres appelee URL entre un navigateur (le client) et un serveur Web.

4.2 XML

(eXtensible Markup Language) Contrairement à HTML, qui est à considérer comme un langage défini et figé (avec un nombre de balises limité), XML peut être considéré comme un métalangage permettant de définir d'autres langages, c'est-à-dire définir de nouvelles balises permettant de décrire la présentation d'un texte. La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données grâce à son extensibilité. Il va permettre de structurer, poser le vocabulaire et la syntaxe des données qu'il va contenir. Les elements d'un document XML sont definit par une description specifique a l'aide de DTD (Document Type Definition) ou avec XML Schema. Certaines applications utilisent XML pour le stockage d'informations grace aux avantages suivantes :

- La nature textuelle des documents XML, leur rend independants des plateformes.
- L'encapsulation des donnees par des balises rend les documents XML lisibles par les humains.
- La validation des documents permet une meilleure communication.

4.3 Le langage WSDL

(Web Services Description Language, est un langage de description de services Web basee sur XML. Son objectif principal est de separer la description abstraite du service de son implementation. Il permet de decrire les services web d'une facon unifiée. Chaque service est vu sous forme d'un ensemble de points d'entrée dans le reseau, capables d'echanger des messages. Le langage WSDL permet de decrire l'interface visible (ou publiee) nomme contrat du service web. Il decritles differents élements du service à l'aide du langage de balises XML :

- Les messages (intervenant lors des echanges) et leurs types associes (pour gerer l'interoperabilité entre les differents intervenants de l'echange).
- Les operations (composées d'un ou plusieurs messages)
- Les liaisons et les ports de communication (permettant de lier les operations a un protocole de transport sous-jacent)
- La description du service lui-meme.

4.4 Protocole UDDI

(Universal Description, Discovery and Integration) est une specification definissant la maniere de publier et de retrouver des informations concernant des services Web. Elle est issue d.une initiative de plus de 200 entreprises dont IBM, Microsoft et Ariba qui avaient pour but de definir une specification commune pour l.enregistrement, l.identification et la decouverte des services en ligne, concus comme des API exploitables depuis un URL. Un annuaire UDDI contient des informations classées par categories sur des entreprises, et les services qu'elles offrent, en associant à ces services des specifications techniques correspondantes. Ces specifications techniques sont generalement definies

en WSDL, ce qui permet d'avoir des informations sur ce que fait le service, comment il communique, et o'u il est heberge. UDDI fournit trois services de bases :

- **Publish :** ce service gere comment le fournisseur de service web s'enregistre lui-meme ainsi que ses services (en utilisant UDDI).
- **Find :** ce service gere comment un client peut localiser le service web desire (cela peut passer par des invocations de services web pour une utilisation automatique par un programme ou par une consultation d'annuaire en utilisant des mots cles).
- **Bind :** ce service gere egalement comment un client peut se connecter et utiliser le service web une fois celui-ci localise.

4.5 Le protocole SOAP

Les communications entre les differentes entites impliquees dans le dialogue avec le service web se font par l'intermediaire du protocole SOAP (Simple Object Access Protocol). Ce protocole est normalise par le W3C. Le protocole SOAP est un protocole leger destiné à l'echange d'informations structurees dans un environnement distribué. Il utilise des technologies XML pour definir une structure de messages pouvant etre echanges sur divers protocoles sous-jacents(p.ex. HTTP, SMTP). Cette structure a ete concue pour etre independante de tout modele de programmation et autre semantique specifique d'implementation. Un message SOAP est une transmission unidirectionnelle entre des noeuds SOAP (d'un émetteur vers un recepteur SOAP), mais les messages SOAP peuvent etre combines par les applications pour implementer les sequences plus complexes d'interactions (requete-reponse, echanges multiples conversationnels).

5 Les challenges pour les systèmes d'information pervasifs orientés services

Les Systèmes d'Information Pervasifs se présentent comme une nouvelle génération des SI dans un environnement pervasif hautement dynamique et hétérogène, dans laquelle l'interopérabilité est devenue une nécessité à tous les niveaux. Ainsi, la vision technologique de l'orientation service permettra de répondre à cette problématique d'interopérabilité. Ces SIP se doivent, de plus, d'être capable d'intégrer, de réutiliser et d'exploiter facilement et d'une manière compréhensible l'ensemble des services qui sont disponibles. Ils doivent fournir des mécanismes de découverte et de composition de services ne nécessitant pas forcément l'intervention d'un utilisateur. Les SIP doivent être conçus dans la seule perspective de répondre aux besoins des utilisateurs de la manière la plus appropriée. Ainsi, une prise en compte des besoins des utilisateurs s'avère un principe fondamental de cette nouvelle génération de SI. La vision proactive peut contribuer dans ce cadre en mettant en avant des informations du contexte d'un service et le contexte courant des utilisateurs (sa localisation par exemple). L'Informatique Pervasive a apporté aux SI de nouveaux défis liés au dynamisme, à l'hétérogénéité et à l'accès ubiquitaire à l'environnement. Elle a apporté également de nouveaux challenges dans l'usage des services au sein des SIP. Ces challenges concernent les aspects suivants :

5.1 La découverte dynamique des services

Les SIP doivent satisfaire au mieux les besoins des utilisateurs mobiles. Ceci nécessite la prise en compte de l'utilisateur et de ses besoins ainsi que de son contexte courant, afin de lui proposer le service le plus approprié en

toute transparence. La découverte de services est ainsi un des défis majeurs des SIP. Ce mécanisme doit assurer un certain niveau de transparence et d'efficacité tout en s'adaptant au contexte et en prenant en considération les besoins de l'utilisateur (afin de garantir une meilleure compréhension de l'utilisation réelle des services). Le challenge ici est de retrouver le mécanisme le plus adéquat pour répondre à tous ces besoins ;

5.2 La composition dynamique des services

Face à l'hétérogénéité des services, les SIP se retrouvent devant le challenge de développer des modèles, des techniques et des algorithmes afin de composer dynamiquement les services hétérogènes et de les exécuter d'une manière transparente tout en prenant en considération l'adaptation aux changements fréquents de l'environnement ;

5.3 La recommandation dynamique des services

Un des challenges des SIP est d'introduire des techniques de recommandation dans la perspective d'augmenter le caractère proactif du système en proposant à l'utilisateur le service le plus approprié qui pourra l'intéresser. La prise en compte du contexte d'un service et le contexte courant nous permet d'anticiper ses besoins, et ainsi rendre les SIP plus proactifs. Le défi est de faire ceci de la manière la plus transparente possible. Il faut donc retrouver les moyens pour observer le contexte dans lequel ils évoluent et sollicitent certains services.

6 Conclusion

Les Systèmes d'Information Pervasifs doivent faire face à l'ensemble des challenges que nous avons évoqués, L'objectif ici est de trouver le mécanisme qui va permettre d'offrir une meilleure pro-activité au système. A ce jour, une majorité de systèmes sensibles au contexte sont simplement réactifs, prenant les décisions en se basant seulement sur le contexte courant. Les recherches dans les systèmes anticipatoires et proactifs sont encore à leurs débuts. Nous pouvons améliorer la transparence des systèmes en anticipant les besoins des utilisaterus et en leurs offrant des services pertinents.

Chapitre IV

Infrastructure Générique pour l'informatique Pervasive

1 Introduction

Le but d'un Système d'Information Pervasif (SIP) est de rendre accessibles les fonctionnalités offertes par le SI à travers un environnement pervasif. Pour ce faire, les SIP doivent faire face à un certain nombre de problèmes, qu'on a cité dans le Chapitre 1. Avec le manque de modèle permettant de prendre en compte tous ces besoins de transparence, d'adaptation à l'environnement et d'adaptation à l'utilisateur d'un SIP. Nous proposons pour lever ces difficultés une architecture générique qui tient en compte les éléments et composants majeurs de base d'un environnement ubiquitaire.

2 Dimension de l'informatique pervasive

En analisant les technologies liées à l'informatique pervasive, nous avons deduit trois dimensions principales de l'informatique ubiquitaire, Figure 4.1 [KB14b] :

- **Réseau sans fils :** représente l'infrastructure de base de tout système pervasif.
- **Réseau de capteurs sans fils) :** embarquée ayant la possibilité de traitement, de stockage, de détection et de communication dans des dispositifs à petite échelle.
- **Dispositifs Pervasif :** Un objet communicant est une entité physique capable de percevoir et de communiquer avec son environnement, avec des utilisateurs et avec d'autres objets quelconques interagissant avec lui.

3 Travaux connexes

Nous faisons une synthèse des travaux existants dans le domaine d'architecture des systèmes ubiquitaires et nous positionnons notre contribution par rapport à ces travaux [KB14a]. Les systèmes d'informatiques ubiquitaires sont au carrefour de plusieurs domaines technologiques (multimédia, réseau, systèmes distribués et embarqués) et préoccupations sociales (confidentialité, fiabilité, sécurité), ce qui les rend particulièrement difficiles à construire et à vérifier. Les architectures actuelles offrent très peu d'abstractions et un support souvent générique et limité. Elles

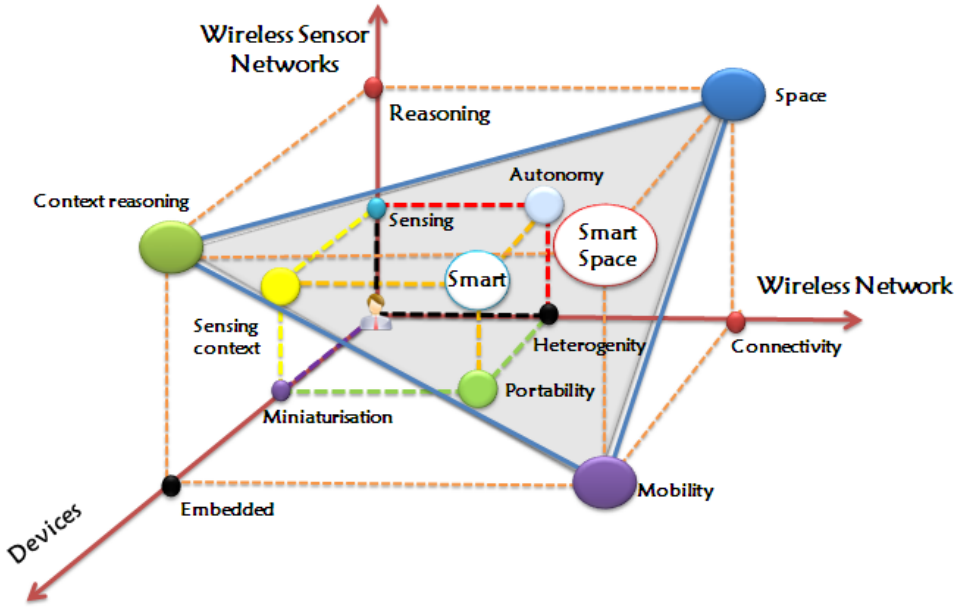

Figure IV.1 – Dimensions principales de l'informatique ubiquitaire.

n'exploitent pas suffisamment cette élévation du niveau d'abstraction pour réellement prendre en compte toutes les spécificités des domaines d'application pour lesquels les systèmes ubiquitaires sont basés.

Plusieurs travaux sur l'architecture des systèmes pervasifs sont proposés. Même s'ils diffèrent, tous ces travaux défendent une évolution des Systèmes d'informations vers des Systèmes d'informations ubiquitaires. Ces approches sont en réalité complémentaires [KB14a]. L'évolution considérable des technologies de communication sans fils couplée aux avancées technologiques matérielles permettent aux dispositifs portables dotés d'une capacité de calcul de devenir disponible partout, [Sat01], [GM04], [ARC10], [AJG12], [ESY04], [SS10], [CV09], [CDB+12], [ZGYR10] adoptent un modèle architecturel à base de ces nouvelles technologies (infrastructure réseaux et dispositifs communicant). Selon [SM03], les capteurs fournissent des informations sur le contexte des systèmes ubiquitaires, ce qui rend ces derniers différents des systèmes classiques. Pour ces auteurs, l'informatique ubiquitaire suppose une étape supplémentaire dans l'évolution initiée avec l'informatique distribuée et poursuivie avec l'informatique mobile. Ils proposent un modèle des environnements ubiquitaires qui inclut quatre domaines : les dispositifs, les réseaux, les intergiciels et les applications. [HTH06] présente un modèle centré utilisateur. Les auteurs [Mir09],[ARC10], [AJG12], [HTH06]] illustrent une thématique de recherche majeure, laquelle met en avant l'importance des espaces intelligents dans lesquels le monde numérique et le monde physique sont liés d'une façon naturelle et transparente pour l'utilisateur. Les travaux [Sat01],[ARC10], [ESY04], [SS10]] intègre le domaine des interfaces utilisateur intelligents qui permetent aux usagers de contrôler et interagir avec des objets de manière intuitive. Enfin [GM04], [ESY04], [HTH06]] aborde la sécurité, qui est un enjeu prépondérant dans la construction d'environnement pervasif.

La plupart des approches proposées dans la littérature sont ou bien spécifiques à une application ou à un domaine particulier ou bien il leur manque le formalisme nécessaire pour la modélisation. Ces approches ne sont pas suffisamment génériques pour être réutilisables dans d'autres applications ou autres domaines. Le cadre général de nos recherches est le domaine de l'informatique pervasive. Notre intérêt porte en particulier sur la proposition d'un nouveau modele architectural générique pour appuyer le développement des systèmes ambiants.

Tableau IV.1 – Synthèses des travaux de recherches, (-) et (+) correspond correspond au degré de prise en charge

	HCI	Dispositifs	Espace Intillegent	Infrastructure	Securité
Satyanarayanan, 2001, [Sat01]	-	+	+	+	+
Debashis SAHA, 2003, [SM03]	+	+	-	+	-
Punnet GUPTA, 2004, [GM04]	-	+	+	+	+
Natalia V,Em, 2005, [ESY04]	+	+	-	+	+
S Hoh, 2006, [HTH06]	-	+	+	+	-
Karlene C, 2009, [CV09]	-	+	-	+	-
Shahid SIDDIQ, 2010, [SS10]	+	+	-	+	+
LAYFOUNI, 2010, [ARC10]	+	+	+	+	-
Jiehan ZHOU, 2010, [ZGYR10]	-	+	-	+	-
Fatma ACHOUR, 2012, [AJG12]	-	+	+	+	-
Marco CONTI, 2012, [CDB+12]	-	+	+	+	+
Notre modéle	+	+	+	+	+

L'informatique ubiquitaire vise à créer un écosystème informatique dans lequel les ordinateurs sont non seulement présents pour les utilisateurs, mais travaillent de plus en plus ensembles. Le tableau 4.1 offre une vue synthétique des propositions qui ont inspiré notre modèle proposé Ces travaux sont complémentaires à notre plate-forme [KB14a].

4 Modéle architectural proposé

Un système informatique pervasif évolue dans un espace ambiant constitué d'une infrastructure de dispositifs de l'environnement physique comprenant les utilisateurs, les objets de la vie de tous les jours, etc. Nous identifiant cinq grands domaines qui constituent un environnement pervasif : l'infrastructure, objets, interfaces, espaces intelligents et la sécurité [KB13b], Figure 4.2.

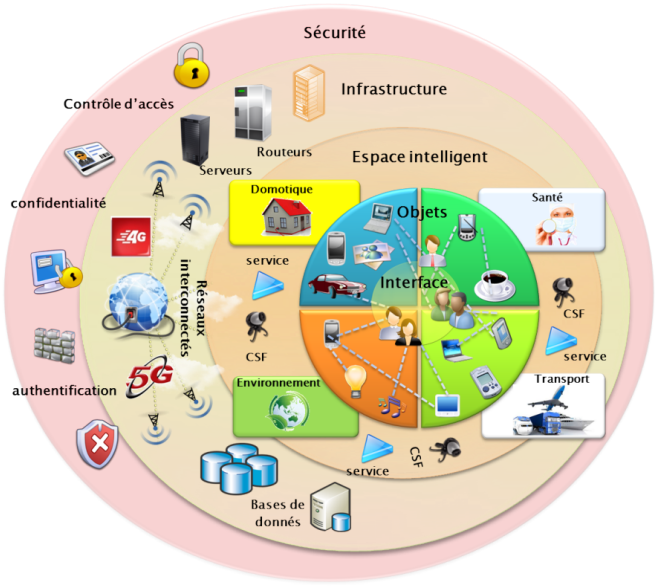

Figure IV.2 – Model architectural proposé.

La couche d'infrastructure intègre l'ensemble technologique pour la conception de systèmes pervasifs, La couche des objets intègre les dispositifs communicants, La couche interface devient une nécessité avec l'émergence de nouvelles technologies d'affichage et d'interaction qui peuvent s'intégrer dans l'environnement pervasif, la couche espace intelligent constituée de dispositifs interconnectés ayant des capacités de perception, d'action et de communication de manière invisible. Enfin, la couche sécurité qui assure la confidentialité et l'intégrité des données. C'est une évolution des SI vers des SI ambiants. Selon cette architecture, un individu peut interagir avec une application à travers un ou plusieurs objets intelligents. L'infrastructure offre le canal de communication, les capteurs sans fil pour collecter les informations nécessaires pour ajuster le comportement du système, les interfaces pour offrir une bonne ergonomie et la sécurité pour la confidentialité et l'intégrité des donnés. La figure introduit les différents éléments du modéle architectural proposé.

4.1 La couche d'infrastructure

Afin de pouvoir réaliser la vision de l'informatique ubiquitaire, nous avons besoin d'une nouvelle infrastructure adaptée au constat qu'autour de chaque utilisateur gravite un ensemble hétérogène d'appareils ayant des capacités de calcul et de connexion. Un environnement ubiquitaire est composé de matériel et modules divers (systèmes d'exploitation et langages de programmation variables). La couche infrastructure est le support de la collaboration transparente entre des équipements par une coopération permanente du réseau d'objets. Ces équipements qui le constituent sont connectés en permanence [KB13b], Figure 4.3.

Figure IV.3 – Infrastructure pervasive.

4.2 Dispositif à accès Pervasif (Objets communicants)

Les objets sont des dispositifs de poche. Dans de tel environnement, ils doivent avoir la notion de communication [Sar12]. Les objets communicants ont différentes utilités ; ils peuvent caractériser l'environnement (notion de capteur),

agir sur l'environnement (notion d'actionneur) et/ou échanger de l'information avec un utilisateur (notion d'interface). Leur disponibilité varie au cours du temps. Ils sont ou bien ajoutées ou supprimées dans l'environnement physique pour divers raisons. Les systèmes ubiquitaires doivent donc considérer la dynamicité des objets communicants en plus de leur hétérogénéité [PR04b]. Dans un avenir proche, les objets communicants vont s'intégrer dans notre vie de tous les jours. Ils collecteront et traiteront des informations de différentes sources et interagiront avec les utilisateurs. Il existe plusieurs propriétés des dispositifs communicants, parmi lesquelles nous citons les suivantes [KB13b], Figure 4.4 :

- **Portabilité :** le dispositif mobile peut être utilisé dans plusieurs emplacements et en mouvement.
- **Interactivité :** le dispositif mobile permet d'échanger des données et collaborer avec d'autres utilisateurs en temps réel.
- **Sensibilité au contexte :** le dispositif mobile peut recueillir des données relatives à l'environnement (à la localisation, à la météorologie.)

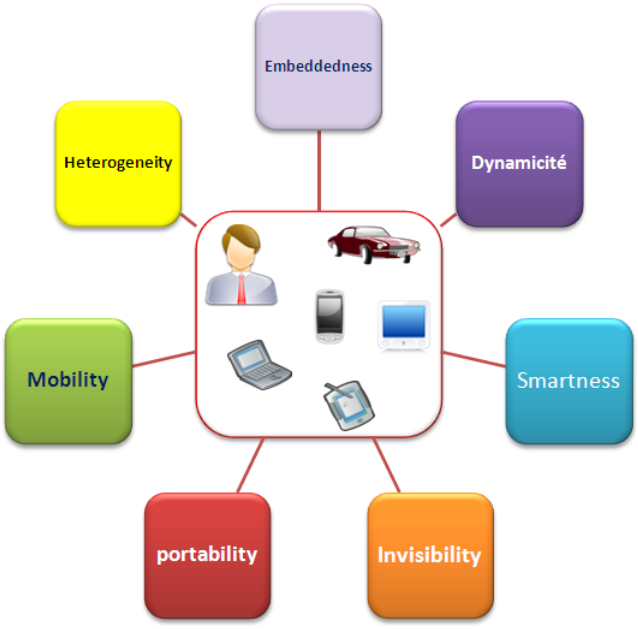

Figure IV.4 – propriétées des objets communicants.

- **Connectivité :** les dispositifs mobiles sont tous munis d'une ou plusieurs formes de connectivité (infrarouge, bluetooth, wifi, GSM, 3G) permettant de les relier à d'autres dispositifs mobiles, à un réseau ou à des dispositifs de collecte de données (capteurs de températures, de tension, etc).
- **Dynamicité :** La disponibilité des objets communicants varie au cours du temps. Ils sont mobiles car connectés sur des réseaux sans fil, ont des ressources limitées, ont des problèmes matériels ou des bogues logiciels les rendant inaccessibles, ou bien sont ajoutées ou supprimées des environnements physiques pour des raisons de maintenance.

4.3 Interface utilisateur

Les interfaces utilisateur intelligentes permettent aux usagers de contrôler et interagir avec des objets d'une manière intuitive et transparente, en ajoutant la notion d'intelligence, c'est à dire la faculté d'analyse du contexte

et l'adaptation dynamique aux situations [Mir09]. L'intelligence ambiante est le prolongement de l'informatique ubiquitaire en y intégrant d'autres domaines, principalement celui des interfaces utilisateur intelligentes[KB13b] , Figure 4.5. Des ordinateurs qui imprègnent l'environnement tout en étant transparents à l'utilisateur et en ajoutant la notion d'intelligence.

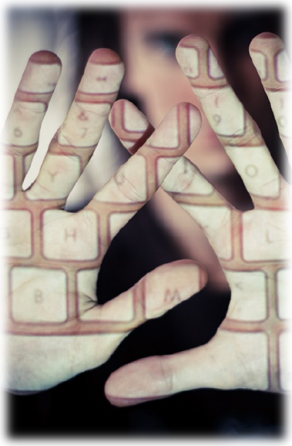

Figure IV.5 – Interface pervasif.

4.4 Espace intelligent

Un espace intelligent est un lieu (physique ou virtuel) d'activités (d'origine humaine ou non) constitué de dispositifs et de fonctions interconnectés, capables de fournir des services attendus avec la valeur attendue [Fon12]. Un espace intelligent, aussi appelé environnement intelligent, peut être identifié comme celui qui est capable d'acquérir et d'appliquer les connaissances sur l'environnement et ses habitants, afin d'améliorer leur expérience. Selon [Sat01], un l'espace peut être un espace clos comme salle de réunion ou un couloir. Les espaces intelligents mélangent deux mondes différents : les environnements physiques et les infrastructures de traitement de l'information, Figure 4.6.

Dans les espaces intelligents [San10], ces deux mondes peuvent s'influencer entre eux : par exemple, l'éclairage d'une pièce peut dépendre du profil électronique d'un utilisateur, et inversement un logiciel embarqué dans un dispositif personnel peut se comporter différemment en fonction de la situation géographique de l'appareil [KB13b].

4.5 Sécurité et confidentialité dans les SIP

La sécurité est un enjeu prépondérant dans la construction d'environnement pervasif. En effet, ces systèmes très ouverts permettent l'accès au système informatique aux personnes présentes dans l'environnement [Bou08]. Cependant, l'accès à certains équipements ou aux données personnelles des utilisateurs doit être fortement sécurisé, [KB13b] Figure 4.7. La mise en place d'un système de sécurité en environnement collaboratif à caractère pervasif, nécessite de considérer les problèmes suivants :

- **L'authentification :**Dans un système de sécurité, l'identification d'un utilisateur donné se fait à travers un processus d'authentification. Cependant, l'évolution technologique, dans sa diversité, offre différents niveaux de sécurité et une variété d'outils permettant de procéder à l'authentification. Ainsi, on peut dénombrer trois principales catégories : L'authentification utilisant ce que l'utilisateur connaît (le code PIN du portable, le login),

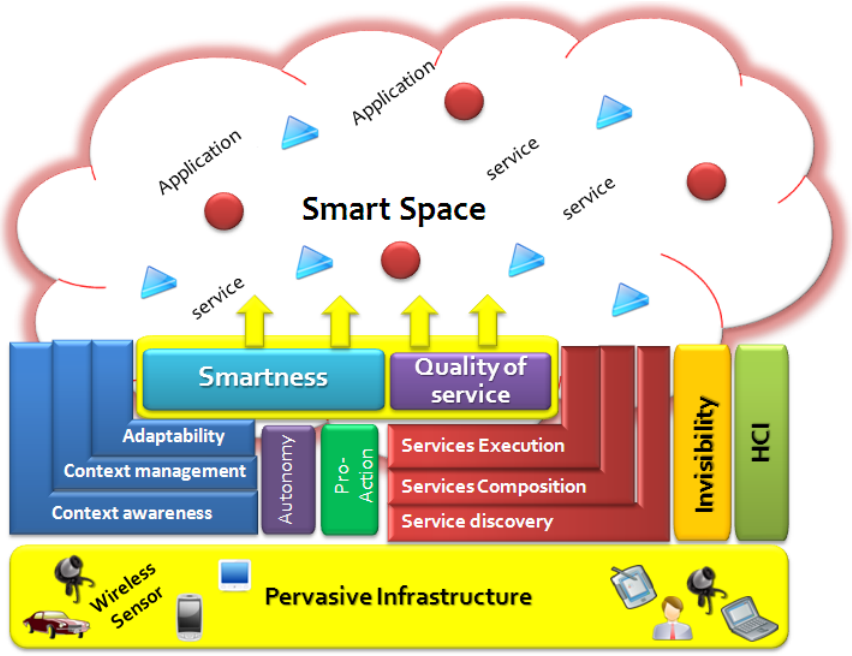

Figure IV.6 – Espace intelligent.

un objet possédé par l'utilisateur (carte bancaire, badge) et l'utilisateur lui-même (authentification biométrique : empreinte digitale, empreinte rétinienne, reconnaissance vocale).

- **Le Contrôle d'accès :** Les environnements pervasifs ont été définis dans le but de s'ouvrir sur un monde dans lequel l'information est accessible n'importe où et n'importe quand. De ce fait, l'administration, l'intégration et la collaboration des différentes politiques de sécurité deviennent plus complexes dans la mesure où ces dernières sont généralement hétérogènes d'un point de vue structurel (rôle, niveau d'accréditation) et sémantique (codification, langue).
- **La confidentialité et la protection de la vie privée :** Le développement de nouvelles technologies contribue à l'évolution de l'informatique ubiquitaire. Cependant, utilisées à mauvais escient, celles-ci peuvent porter atteinte à la vie privée des usagers. En effet, un utilisateur a une perception très limitée des risques potentiels issus des différents équipements qu'il peut embarquer. Prenons l'exemple du passeport biométrique qui contient un tag RFID et qui peut communiquer l'identité de son propriétaire juste en le passant devant un lecteur. Une personne peut ainsi être reconnue à n'importe quel endroit, en passant par exemple par le centre commercial ou en prenant le train. Les informations stockées dans les RFID ou toutes autres informations échangées entre l'utilisateur(ou ses dispositifs) et l'environnement sont en général signées (ex : les certificats). Ces documents numériques sont la propriété de l'usager qui doit avoir le droit de disposer des différentes informations fournies par ces certificats de manière sélective et contextuelle.

5 Implémentation et Simulation

Dans cette thèse, nous proposons une approche visant à concevoir et développer les applications ubiquitaires avant leur déploiement en environnements réels et virtuels. Cette approche a donné lieu au développement d'une

Figure IV.7 – La sécurité dans les systémes pervasifs.

boite à outils dédiée à l'exécution des applications ubiquitaires [KB14a]. Les applications ubiquitaires sont testées dans un simulateur d'environnements ubiquitaires, appelé **Generic Architecture for Pervasive Computing Simulator :GA4PCSim** [KB13b].

5.1 GA4PC^{Sim}

L'Environnement expérimental, appelé GA4PCSim est constitué de divers dispositifs, comme autant de services découvrables [KB14b]. Ces dispositifs peuvent être soit des dispositifs virtuels (objets d'une scène 2/3D dans laquelle l'utilisateur est immergé), soit des dispositifs réels portés par l'utilisateur ou présents dans son environnement, Figure 4.8. Tous les équipements de GA4PCSim, réels ou virtuels, sont donc basés sur une interface de type Web Service classique ou pour dispositifs. GA4PCSim mis en œuvre repose sur trois grandes classes d'équipements :

- **Environnement réel de l'utilisateur :** des dispositifs sans-fil présents dans l'environnement, tels que des capteurs (luminosité, température, accéléromètre, ...) et actionneurs (télé-relais, ...),
- **Centré utilisateur :** dispositifs d'interaction : joystick, téléphone portable, PDA,
- **Environnement simulé :** sous forme d'une scène virtuelle 2/3D, des dispositifs virtuels UPnP associés à des objets de la scène.

Un tel environnement est un cadre idéal pour l'évaluation de nouvelles applications de l'informatique pervasive telles que les usages des ordinateurs portés. Cette plateforme est donc tout particulièrement adaptée à l'étude de l'informatique ubiquitaire sensible au contexte. GA4PCSim permet de gérer l'apparition et la disparition des Web Services pour Dispositifs ; il gère l'instanciation et la destruction automatiquement des dispositifs UPnP. Il gère les variations de l'infrastructure sans intervention directe de l'utilisateur. Le designer offre également la possibilité de manipuler ces dispositifs au travers de leur interface. L'architecture de la Toolbox repose sur [KB14b] :

- **Une approche à composant** est adoptée à cause de la réutilisabilité. On peut remplacer un composant tout comme on installe ou on remplace une ampoule dans le monde réel ou un équipement électronique dans un

Figure IV.8 – Simulateur des espaces intelligents

espace intelligent.

- **Une approche à service** est retenue pour la découverte et la disparition dynamiques d'entités. Un service peut être implémenté et fourni par un composant quelconque pour être utilisé par un autre composant. Autrement dit, dans un espace intelligent, un service est assuré par le composant.
- **Une communication événementielle** est retenue par la nécessité de construire un système proactif au changement de contexte. un espace intelligent, qui est constitué d'une multitude de dispositifs computationnels, n'est pas un système centralisé. En réponse au requis de distribution (et d'hétérogénéité), les dispositifs de l'espace intelligent sont supposés répondre au protocole UPnP.

5.2 Simulation

Comme le montre le plan de la Figure 4.9, un appartement est équiper de capteurs et de dispositifs communicants natifs UPnP. C'est le cas par exemple des équipements audio, volets, Smartphones et les lumières.

La Figure 4.10 represente la simulation virtuelle d'une maison intelligente qui comprend une chambre, une salle de bain, une cuisine, une salle à manger et un salon. L'utlisateur est reconnue par des détecteurs de mouvements infrarouges, des capteurs RFID et des cameras installés dans la maison. Une fois l'interessé entre dans la maison, divers services sont invoqués sans son intervention :

- La lumiere est allumé dans sa chambre.
- Quand il se deplace vers le Hall, son portable se met en mode vibreur.
- Enfin, une fois il entre dans la cuisine le dispositif se met à chauffer son café.

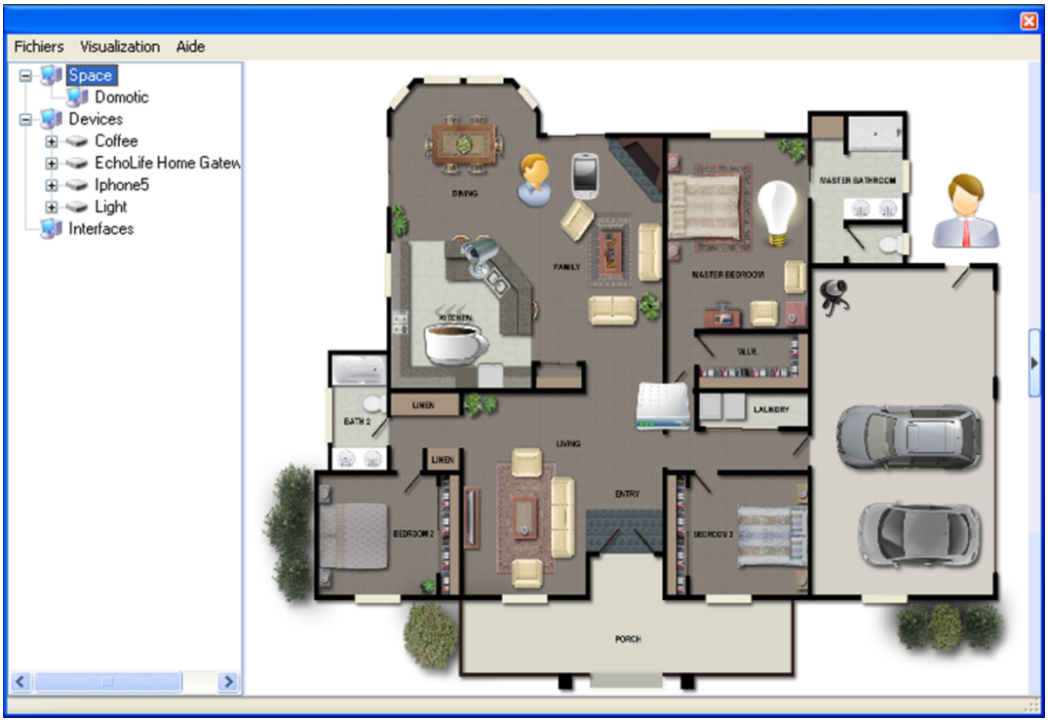

Figure IV.9 – Espace intelligent : Maison intelligente.

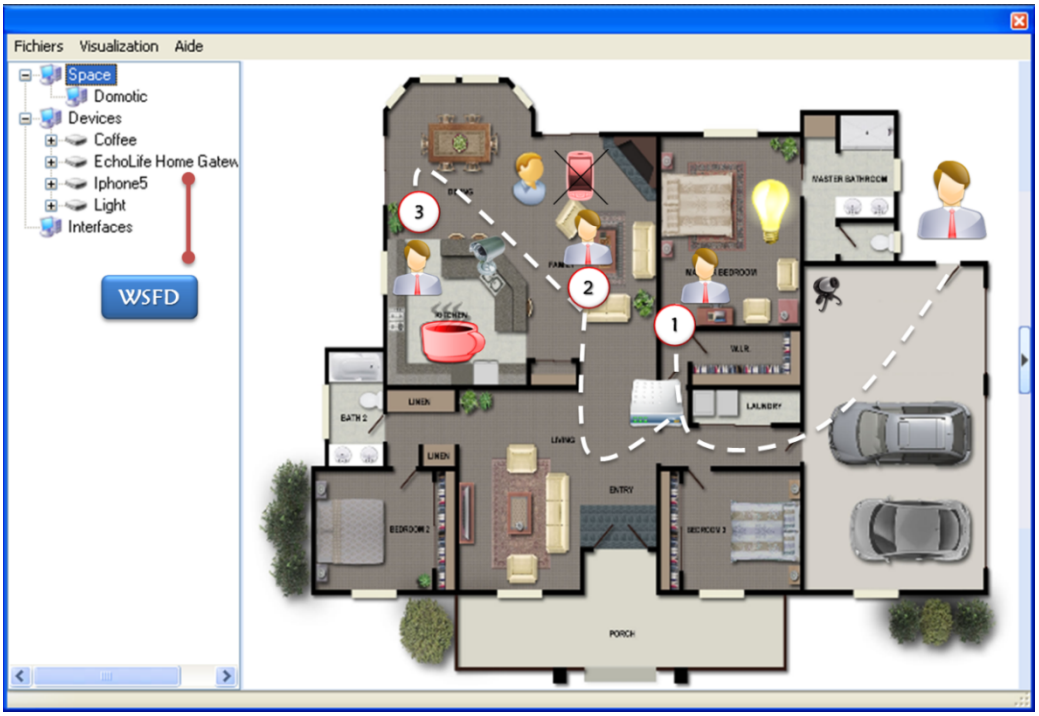

Figure IV.10 – Simulation de la Maison intelligente.

6 Conclusion

Dans ce chapitre, Notre contribution se résume en un modèle architectural générique qui tient en compte les éléments de base d'un environnement ubiquitaire. Nous avons identifié cinq grands domaines qui constituent un environnement pervasif : infrastructure, objets, interfaces, espaces intelligents et sécurité. La couche d'infrastructure intègre l'ensemble technologique utilisé pour la conception des systèmes pervasifs, La couche des objets intègre les dispositifs communicants, La couche interface permet aux usagers de contrôler et interagir avec des objets d'une manière intuitive, la couche espace intelligent repose sur des capacités de perception, d'action et de communication. Enfin, la couche sécurité pour garantir la confidentialité des informations. Selon cette architecture, Un individu peut interagir avec une application à travers un ou plusieurs objets intelligents. L'infrastructure offre le canal de communication, les capteurs sans fil Pour collecter les informations nécessaires pour ajuster le comportement du système, les interfaces pour offrir une bonne ergonomie et la sécurité pour la confidentialité et l'intégrité des donnés.

Nous avons développé un simulateur pour tester le comportement des applications ubiquitaires à l'exécution. Notre approche permet de tester des applications dans des environnements hybrides constitués d'entités réels et simulés.

Chapitre V

Modélisation du contexte pour l'informatique Pervasive

1 Introduction

Les environnements pervasifs sont peuplés de périphériques informatiques hétérogènes hébergeant des services sensibles aux informations contextuelles. Toute fois ces informations sont hétérogènes dans leurs représentations et récupérées de sources distribuées dans l'environnement. La partage d'information dans de tel environnement nécessite donc de définir un modèle uniforme de représentation d'informations qui sera partagé par l'ensemble des services.

Dans ce chapitre, nous présentons notre modèle de représentation de contexte. Nous utilisons une approche basée sur les ontologies. De plus, nous exposons une approche de raisonnement à base de règles SWRL, pour dériver des informations concernant le contexte. Dans l'état de l'art, nous avons pu constater diverses notions de contexte et leurs modèles. Ils partagent des concepts communs et peuvent hériter d'une notion de contexte plus générique.

2 Définition d'une ontologie

Historiquement, le terme Ontologie a été défini en philosophie comme une branche de la métaphysique qui s'applique à l'être en tant qu'être [BOU10]. En effet, ce terme grec est composé des mots " ontos" et " logos " qui veulent dire respectivement l'essence de l'être. Au début des années 90, des chercheurs en intelligence artificielle se sont intéressés à cette notion pour la formalisation des connaissances.

Du point de vue informatique, différentes définitions de l'ontologie ont été données. La plus référencée " Une ontologie est une spécification formelle et explicite d'une conceptualisation partagée ", la conceptualisation signifie un modèle abstrait d'un certain aspect du monde, prenant la forme d'un algorithme qui détermine des relations d'instance [Gru93] : l'individu i est une instance de la description de concept C si et seulement si i est toujours interprété comme un élément de C. Le terme explicite signifie que les concepts utilisés ainsi que les contraintes sur leur utilisation sont définis d'une manière claire et précise. L'adjectif formelle se réfère au fait que l'ontologie doit être lisible et compréhensible par les machines, cela est nécessaire pour qu'elle puisse être munie de capacités de raisonnement, permettant de décharger les utilisateurs d'une partie de leur tâche d'exploitation et de combinaison des ressources du Web.

3 Propriété d'une ontologie

Même si le besoin de développer une ontologie est très varié et dépend du domaine d'application, nous pouvons facilement énumérer un certain nombre d'utilités, notamment :

- **La connaissance du domaine :** Les ontologies permettent la modélisation des connaissances dans un domaine particulier, dans lequel opère le système à développer.
- **La communication :** les ontologies assurent une communication fiable et hétérogène entre personnes et machines (agents logiciels ou organisations) du fait qu'elle permet de mettre en place un langage ou un vocabulaire conceptuel commun [UGUG96].
- **L'interopérabilité :** La représentation explicite des connaissances dans un domaine donné sous forme d'une ontologie, permet à son tour une plus grande réutilisation, un partage plus large et une interopérabilité plus étendue[UGUG96].
- **L'aide à la spécification des systèmes :** La représentation conceptuelle des éléments du domaine, permet aux systèmes de réaliser des raisonnements logiques qu'on appelle inférences, et de sortir avec des conclusions capables d'aider l'utilisateur ou le gestionnaire dans ses décisions.
- **L'indexation et la recherche d'information :** Dans le web sémantique, les ontologies sont utilisées pour indexer et décrire les ressources utilisées. Cela permet une plus grande précision dans les résultats des recherches ou d'assignation des ressources.

4 Langages de présentation d'une ontologie

Dans le contexte du Web Sémantique, plusieurs langages ont été développés. La plupart de ces langages reposent sur XML ou utilisent XML comme syntaxe. Nous allons présenter brièvement certains langages principaux XML, RDF(S) et OWL.

4.1 Unicode URI et XML

L'URI est un protocole simple et extensible pour identifier, d'une manière unique et uniforme, toute ressource sur le web [BOU10]. Il s'agit d'un aspect central de l'infrastructure. Le langage XML est un langage de description et d'échange de documents structurés, issu de SGML (Standard Genralized Markup Language). XML permet de décrire la structure logique de documents à l'aide d'un système de balises permettant de marquer les éléments qui composent la structure et les relations entre ces éléments.

La connaissance de la structure d'un document XML permet notamment de vérifier la validité de ce document. Un fichier de description de structure (XML Schema Description, ou fichier XSD) est donc lui-même un document XML. Les espaces de nommage XML offrent une méthode simple pour qualifier les noms des éléments et les attributs utilisés dans des documents XML, en associant ceux-ci avec des espaces de nommage désignés par des références d'URI.

4.2 RDF et RDF Shema

RDF [LS99] est un modèle de données pour les objets (ressources) et les relations entre eux fournissant des sémantiques simples pour ce modèle de données qui peuvent être représentés en XML. RDF permet de représenter des métadonnées à propos des ressources (identifiées par des URI) du web. La construction de base en RDF consiste

en un triplé d'éléments (Ressource, Propriété, Valeur), qu'on appelle déclaration RDF. Par analogie, un triplé RDF est similaire à la déclaration <sujet - prédicat - objet>.

- **Ressource (Sujet)** : Cela peut être n'importe quel objet référencé par une URI, qu'il concerne le web (Page HTML, document PDF, fichier multimédia), ou non (Personne, Région, Etc.).
- **Propriété (prédicat)** : Critère, caractéristique, attribut ou relation qui peut décrire la ressource (titre, couleur, taille, auteur, etc.).
- **Valeur (objet)** : C'est la valeur qui sera affectée à la propriété de la ressource. Cette affectation peut être soumise à certaines restrictions.

4.3 OWL

Le langage OWL est un composant de l'activité Web Sémantique qui vise à rendre les ressources Web plus accessibles aux processus automatisés en ajoutant des informations qui décrivent le contenu Web [BOU10]. Le langage d'ontologie Web OWL définit des ontologies Web. Une ontologie OWL peut contenir des descriptions de classes, de propriétés et de leurs instances. Pour une telle ontologie donnée, la sémantique formelle OWL indique comment déduire ses conséquences logiques, c'est-à-dire les faits qui ne sont pas littéralement présents dans l'ontologie mais déduits par la sémantique. OWL peut être défini en trois sous langages complémentaires proposant une expressivité croissante, chacun conçu pour des communautés de développeurs et des utilisateurs spécifiques : OWL Lite, OWL DL, OWL Full, Voir chapitre 2.

5 Modélisation à base d'ontologies pour l'informatique pervasive

Parmi les travaux s'intéressant à une modélisation à base d'ontologies dans les environnements pervasifs, [CFJ03], proposa un courtier de contexte (Context Broker Architecture ou CoBrA). Il définit une collection d'ontologies appelée COBRA-ONT pour la modélisation de contexte dans un environnement d'une salle de rencontre intelligente. [CPFJ04] ont proposé une ontologie partageable nommée SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications) pour favoriser le partage de connaissances et le raisonnement sur le contexte, [WZGP04] ont proposé une ontologie du contexte CONON (CONtext ONtology) codée en OWL pour la modélisation du contexte dans un environnement diffusé et pour le support de raisonnement logique sur le contexte. CONON fournit une première ontologie du contexte supérieur qui capte les concepts généraux du contexte de base, et fournit aussi l'extensibilité pour ajouter des ontologies spécifiques à un domaine particulier d'une manière hiérarchique. [SLP04] [GPZ05] ont introduit l'approche CoOL (Context Ontology Language). Celle-ci utilise les ontologies et le modèle de contexte ASC (Aspect-Scale- Context), [GWPZ04]ont proposé une approche de modélisation du contexte à base d'ontologies dans leur architecture SOCAM (Service-Oriented Context-Aware Middleware). Elle est composée de deux parties : une ontologie supérieure qui capte les connaissances générales du contexte du monde physique d'un environnement diffusé, [PdBW+04] ont proposé une ontologie du contexte adaptable et extensible pour la création d'infrastructures sensibles au contexte, l'ontologie est exprimée en OWL. Elle est constituée de quatre ontologies de base : utilisateur, environnement, plateforme et service. [CK05] supporte la decouverte dynamique des services. [RHC+02] le projet GAIA, ont également proposé l'utilisation d'ontologies pour décrire les informations contextuelles des milieux. Dans GAIA, l'ontologie propre aux milieux est utilisée dans ces opérations d'adaptation des applications aux profils des appareils. [GKPMD13] ont proposés une approche sémantique constituée d'un modèle personnalisable exprimé comme un ensemble d'ontologies qui convient à de nombreux types d'organisations différentes.

La plupart des modèles ontologiques proposés n'offrent pas une description complète des informations contextuelles. Dans le but d'offrir une ontologie extensible et réutilisable. Notre objectif est de proposer une ontologie de service qui couvre tous les aspects du contexte dans un systeme d'information pervasif, suffisamment générique et facile à utiliser dans diverses applications sans efforts considérables d'adaptation.

6 Modélisation du contexte pour l'informatique pervasive

La notion de contexte (chapitre 2) est un élément essentiel pour la réalisation de systèmes sensibles au contexte. Or, tout SIP doit également être sensible au contexte pour pouvoir s'adapter à l'environnement pervasif dans lequel évoluent ses utilisateurs. Une représentation formelle du contexte est nécessaire afin d'en définir les contours et d'identifier les informations pertinentes, qui influenceront effectivement le comportement du système.

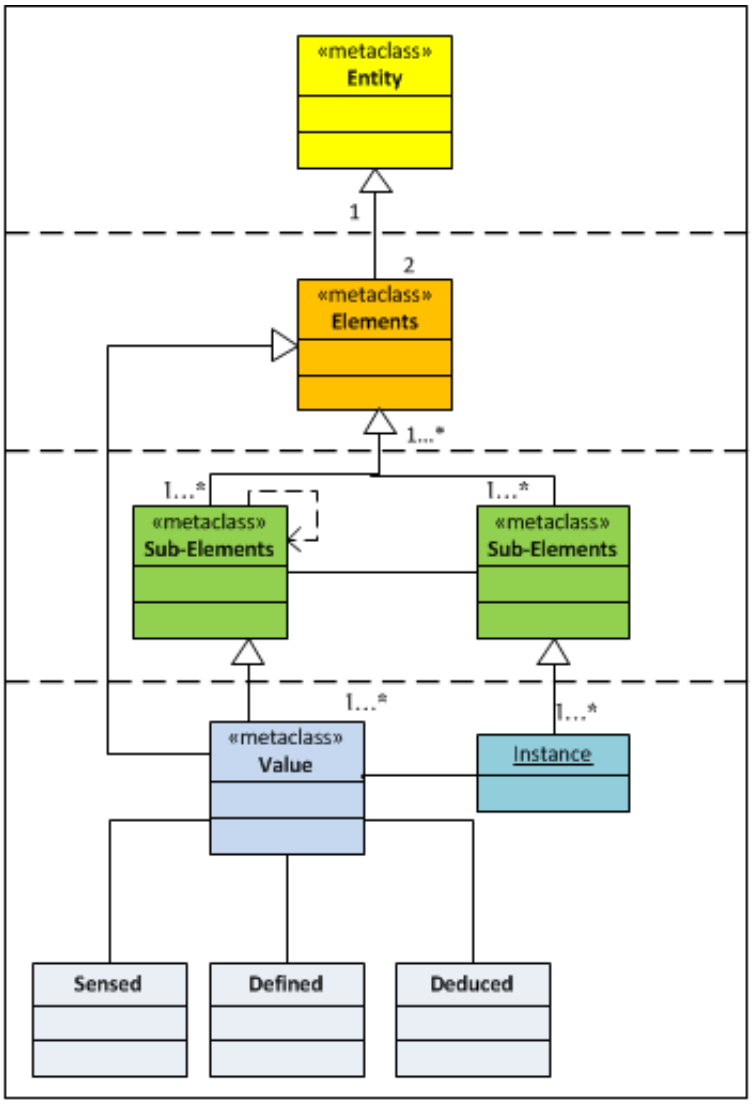

Figure V.1 – Meta Modèle Du Contexte.

D'après l'analyse des différentes définitions et approches de modélisation de contexte, que nous avons exposées dans le chapitre 2. Nous pouvons dégager certains éléments clés et communs, qui caractérisent l'immense majorité de ces modèles. En se basant sur la définition référence de Dey," toute information qui peut être utilisée pour caractériser la situation d'une entité ", trois concepts clés peuvent être dégagés :

- **entité :**qui fait référence à un domaine.
- **élément :** qui fait référence à quelque chose observable dans l'environnement.
- **toute information** qui est présentée comme tout attribut observé de cette entité.

Nous visons une notion de méta-modèle flexible [KB14b]. Ce méta-modèle permet de décrire comment les informations de contexte générales peuvent être modélisées. La généricité, visée par le modèle, fournit une vue des structures de base qui peuvent être utilisées lors de la définition d'un modèle de contexte spécifique à un domaine donné. Dans notre modélisation nous avons défini deux dimensions essentielles :

1. **La dimension espace ou environnement**, elle exprime des informations concernant la localisation (par exemple, les coordonnées géographiques, l'adresse postale, etc., les objets fixes à proximité). Elle inclut aussi des informations liées à l'instant de l'action (par exemple, la date, le jour de la semaine, etc.).
2. **La dimension infrastructure** fait référence aux aspects computationnels et non computationnels. L'aspect non computationnel est lié aux personnes présentes à l'intérieur de l'environnement (par exemple, l'utilisateur du système, son réseau social. L'aspect computationnel est lié aux dispositifs, réseaux, application et services.

La Figure 5.1 illustre une représentation graphique de notre ontologie à lŠaide du plug-in OWLViz [KB14b]. Ce dernier est un outil permettant la visualisation des ontologies OWL. Le concept Entité (Context Entité) est composé de plusieurs éléments contextuels (Context Elément). Le contexte peut également décrire la situation de ses sous-éléments pouvant être instanciés et permet de déterminer une valeur soit acquise dynamiquement à travers des capteurs, soit définie statiquement au départ ou dérivée ensuite à travers un processus d'interprétation. Nous avons choisi de segmenter les éléments du contexte en deux concepts de niveau supérieur : espace et infrastructure. Cette segmentation est représentée par la spécialisation du concept Context Elément.

Par exemple, dans une modélisation de contexte sous forme d'ontologie, on souhaite observer le sous élément de contexte " profil " de l'élément " utilisateur ". Cet élément de contexte se caractérise par d'autres sous éléments comme " âge ", " expérience ", etc. De ce fait, notre méta-modèle de contexte répond aux faiblesses des autres modèles de contexte à gérer l'aspect hiérarchique entre les éléments de contexte. Par exemple " LocalisationParGPS " et " LocalisationParAdresse " sont représentés comme deux éléments de contexte totalement distincts, alors qu'en réalité ils ne sont que des éléments composants de l'élément de contexte " localisation ". Ceci peut entrainer certaine confusion lors du traitement du modèle de contexte. Chaque élément de contexte est composé de plusieurs valeurs observées.

Nous proposons un modèle de contexte soutenu par une ontologie qui prend en considération tous les éléments qu'on a mis en avant lors de notre méta-modélisation de contexte [KB14b]. Les concepts de base de notre ontologie de contexte sont illustrés dans la Figure 5.2. L'ontologie de contexte que nous proposons se base sur le langage d'ontologie Web OWL DL (Description Logics). Cette version du langage OWL est suffisamment expressive, ainsi tous les concepts héritent des propriétés du concept " owl : Thing " qui représente l'entité de la description OWL. L'élément de contexte fait référence aux entités : Espace et Infrastructure. De plus, cette ontologie tourne autour des sous éléments de contexte, qui correspondent à toutes entités statiques ou dynamiques, tels que dispositifs, localisation, etc.

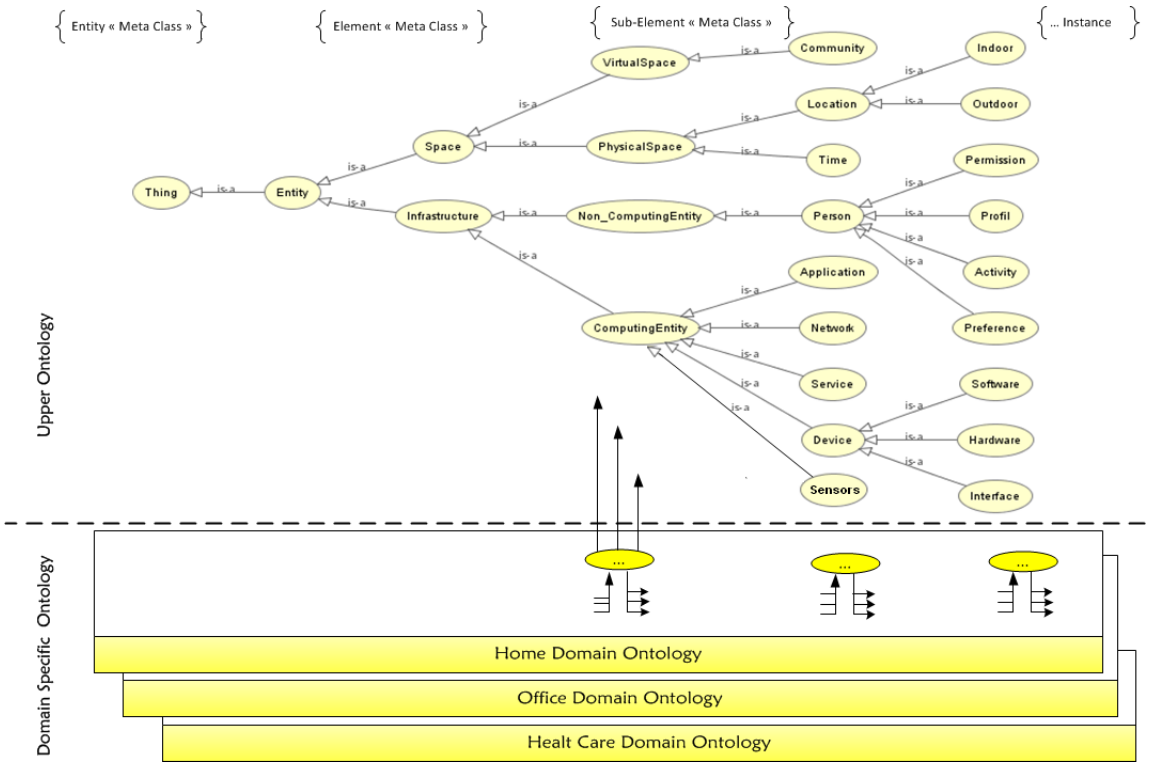

Figure V.2 – Ontologie multi-niveaux.

Pour l'implémentation de l'ontologies et pour la création des règles, nous avons choisi d'utiliser l'environnement de développement d'ontologie Protégé-2000.[KFNM04], Figure 5.3.

Afin de fournir une ontologie extensible qui est bien structurée, nous utilisons une ontologie multi-niveaux [KB14b]. Cette ontologie est conçue pour être facilement extensible par des sous-ontologies, dépendantes du domaine, en fonction du système dans lequel le modèle s'applique. L'utilisation d'ontologies offre une description sémantique particulièrement riche et fournit de nouvelles perspectives pour les mécanismes d'adaptation grâce aux différentes possibilités de raisonnement.

A la Figure 5.2, nous représentons, l'ensemble des concepts que nous avons identifiés comme couramment utilisés dans la littérature. Par exemple, " GPS " représente un héritage du concept " Outdoor " qui à son tour représente un héritage du concept " localisation " qui représente lui-même une spécification du concept " espace ". Cette ontologie multi-niveaux est structurée comme suit :

- **Ontologie générique (Niveau supérieur) :** décrit les informations de base de contexte dans un environnement pervasif. Ces informations sont assez génériques et communes à plusieurs domaines. A ce niveau, les concepts de base sont définis. Deux concepts de niveau supérieur sont ainsi définis, Figure 5.2 : espace et infrastructure. Ces concepts sont organisés autour du concept " Entité " qui représente le point d'entrée pour déclarer l'ontologie de haut-niveau.
- **Ontologie spécifique du domaine (Niveau inférieur) :** comporte des informations de contexte spécifiques à un domaine d'application qui varie d'un domaine à un autre. Par exemple, le concept " localisation " de l'ontologie de haut niveau peut être spécifié en " localisation " qui est de même spécifiée en " bâtiment ", " chambre ", entre autres. Dans un autre domaine, ce concept " localisation " peut être spécifié en " localisation

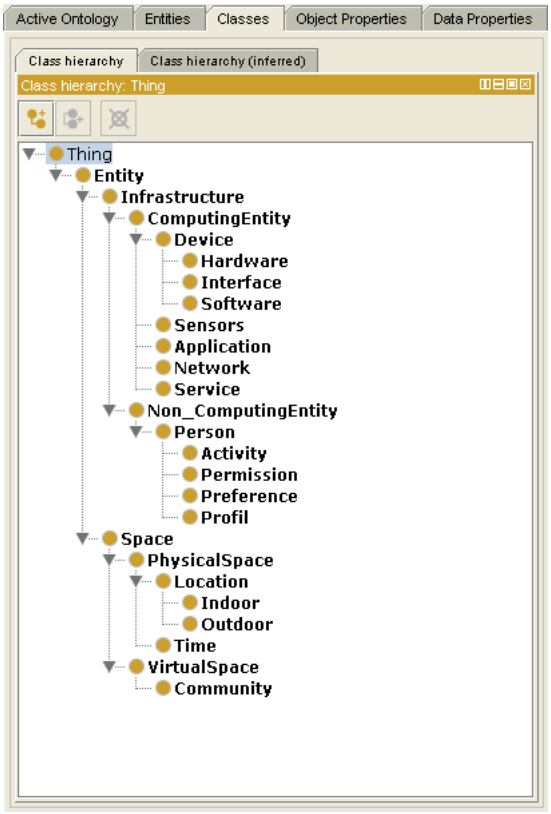

Figure V.3 – Segment de listes des Classes de notre modele de contexte sous Protégé-2000.

GPS " qui a comme propriétés : " longitude ", " latitude " et " altitude ".

La Figure 5.4 illustre un exemple pour la caractérisation du contexte selon la définition de Dey [DAS01]. Le contexte décrit les éléments du système d'information hospitalier. Un exemple d'observation du contexte est présenté. Nous pouvons interpréter cette instanciation du modèle comme étant : Le médecin cardiologue, localisé sur un lieu (hôpital) précisément un bureau. Le Docteur utilise le dispositif Iphone5 pour interagir avec le système.

7 Raisonnement sur le Contexte

7.1 SWRL

Le langage OWL permet de spécifier des ontologies sous forme de classes et de propriétés, cependant son expressivité est limité lorsqu'il s'agit d'exprimer des relations complexes. Il ne permet pas d'exprimer des relations de composition entre propriétés. Par exemple il est impossible de spécifier la relation Oncle comme étant la composition de la relation père et frère. Afin de pallier ce problème d'expressivité , une solution consiste à étendre le langage OWL avec un langage de règle permettant d'exprimer ses relations. Nous nous intéressons particulièrement au langage SWRL qui permet d'étendre OWL avec des règles logiques, il s'impose comme un standard.

Le raisonnement sur les ontologies est réalisé en appliquant un ensemble de règles aux faits contenus dans une base de connaissance. L'exécution des règles est effectuée par un moteur d'inférence. Il accepte en entré un ensemble de faits provenant d'une base de connaissance. Le mécanisme de résonnement permet au moteur d'inférer de nouveaux

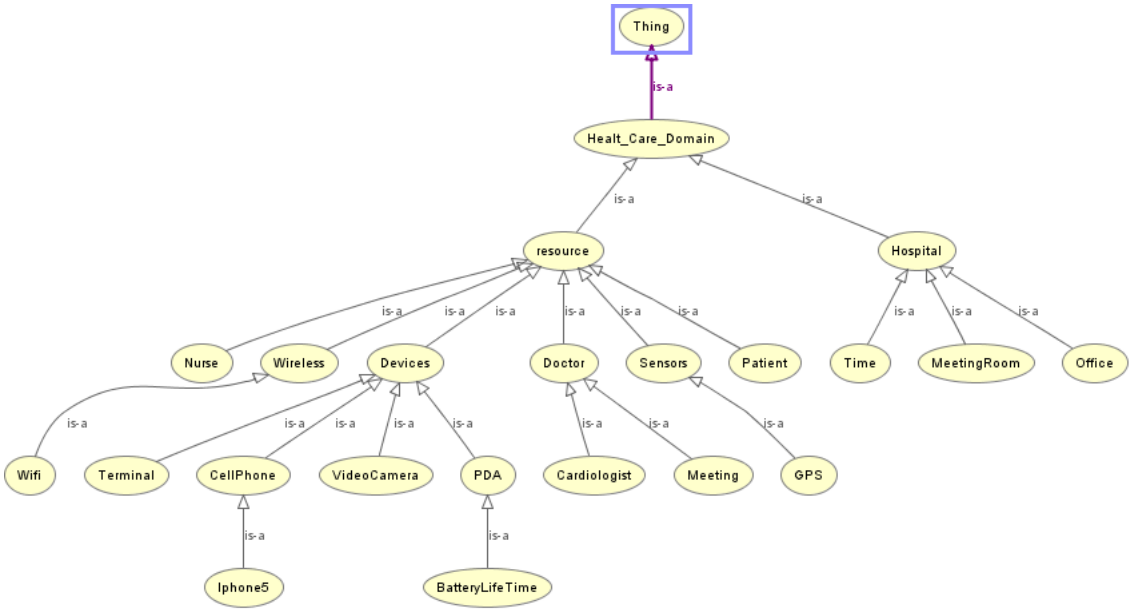

Figure V.4 – Ontologie spécifique au domaine de la santé.

faits à partir des fiats qui sont stockés dans la base de connaissance. Les nouveaux faits sont a leurs tour stockés dans une base de connaissance et sont utilisés pour exécuter de nouvelles règles, Figure 5.5.

7.1.1 Définition du Langage SWRL (Semantic Web Rule Language)

[HPSB+04] est un langage de règles pour le web sémantique combinant le langage OWL-DL et le langage RuleML (Rule Markup Language (Unary/Binary Datalog)). SWRL est un langage qui enrichit la sémantique d'une ontologie définie en OWL. SWRL permet contrairement à OWL de manipuler des instances par des variables (?x, ?y, ?z). SWRL ne permet pas de créer des concepts ni des relations. Il permet simplement d'ajouter des relations suivant les valeurs des variables et la satisfaction de la règle.

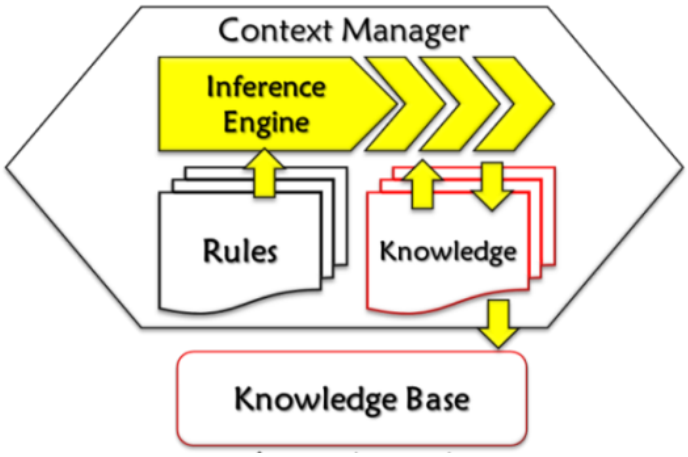

Figure V.5 – Swrl.

OWL permet de spécifier des propriétés algébriques pour les relations. Mais il lui manque surtout des possibilités pour encoder des connaissances plus générales, relatives en particulier à la composition des relations. Ainsi, SWRL constitue un pas vers le rapprochement entre les langages OWL et les règles logiques. SWRL intègre une syntaxe abstraite de haut niveau pour les règles de Horn dans le langage OWL. OWL et SWRL sont les langages fondamentaux du Web sémantique. OWL est un langage d'ontologie développé pour construire des ontologies fournissant un haut niveau d'expressivité pour le contenu du Web. SWRL est basée sur OWL, il garde la puissance d'OWL DL mais au prix de la décidabilité. SWRL permet aux utilisateurs d'écrire des règles pour raisonner sur les individus et pour en déduire de nouvelles connaissances sur ces individus.

7.1.2 La syntaxe

La syntaxe SWRL définit une règle qui est une relation d'implication entre un antécédent (corps) et un conséquent (tête) [HPSB+04]. Si les conditions spécifiées dans l'antécédent sont vérifiées c'est-à-dire si l'on a pu démontrer la partie corps, alors les conditions spécifiées dans le conséquent le sont aussi.

L'antécédent et le conséquent d'une règle sont des conjonctions d'atomes. Les atomes peuvent avoir les formes :

- **C(x)** où C est une description OWL, x est soit une variable, un individu OWL ou encore des valeurs de données (datavalues) de OWL.
- **P(x,y)**, où P est une propriété OWL (Object-property ou data-type-property), x est soit une variable ou un individu OWL et y est soit une variable, un individu OWL ou encore des valeurs de données de OWL.
- **sameAs(x,y)**, differentFrom(x,y) où x et y sont des variables ou des individus OWL.

Le fonctionnement d'une règle est basé sur le principe de satisfiabilité de l'antécédent ou du conséquent. Pour une règle, il existe trois cas de figure :

- L'antécédent et le conséquent sont définis. Si l'antécédent est satisfait alors le conséquent doit l'être aussi.
- L'antécédent est vide, cela équivaut à un antécédent satisfait ce qui permet de définir des faits.
- Le conséquent est vide, cela équivaut à un conséquent insatisfait, l'antécédent ne doit pas être satisfiable.

7.1.3 Moteurs d'inférences sur SWRL

L'exécution des règles SWRL nécessite un moteur de règles [HPSB+04]. Il applique les faits sur les règles pour déduire des nouveaux faits. Par exemple, dans une ontologie on peut représenter la relation entre un oncle et son neveu à partir des relations père-fils et frère-frère. La règle SWRL qui exprime cette relation est représentée dans l'exemple de la Figure 5.6.

```
Personne(?x) ∧ Personne(?y) ∧ Personne(?z)
∧ pere(?x,?y) ∧ frere(?x,?z) → oncle(?z,?y)
```

Figure V.6 – Exemple de règle SWRL.

Les éléments x, y et z, qui sont précédés d'un point d'interrogation, sont des variables qui représentent des individus. Cette règle veut dire que, si l'on a trois individus appartenant à la classe Personne appelés x, y et z, que x est le père de y et que x a un frère z, alors z est l'oncle de y, une instance (z, y) de la propriété oncle est crée dans l'ontologie comme présentée dans la figure 5.6.

Il existe plusieurs moteurs d'inférence permettant l'exploitation des règles SWRL. Dans notre Systeme nous avons choisi l'utilisation du moteur d'inference SWRLTab qui offre un environnement de developpement dans l'outil Protege pour travailler avec des regles SWRL [KB14b]. L'editeur SWRL est accessible sous forme d'onglet et permet l'edition et l'execution des regles SWRL.

Nous utilisons SWRLJessTab un plug-in à SWRLTab dans Protege qui prend en charge l'exécution des règles SWRL en utilisant le moteur de règles JESS. Il fournit une interface graphique pour interagir avec le SWRLJessBridge. Ce module permet d'extraire les règles SWRL, à partir de note ontologie de contexte formalisée en langage OWL, comme il fournit une API permettant l'ajout des nouvelles connaissances déduites à notre ontologie.

7.2 Application aux Environnements Pervasifs

Les environnements pervasifs sont dynamiques par nature. Les dispositifs entrent et sortent de l'environnement ce qui entraine une disponibilité dynamique des services, lorsqu'un service annonce sa présence à l'infrastructure par le biais d'un protocole de découverts de service (UPnP dans notre cas). L'infrastructure intègre le nouveau service dans le modèle de contexte, Le modèle de contexte doit refléter cet état et en particulier la disponibilité des services pertinents. La gestion du contexte est relative aux informations de lieux, de personnes et de paramètre physiques qui sont extrait de la base de connaissance et d'inférer des informations de plus haut niveaux. Prenant l'exemple suivant : Intérieur (?lieu) contient (?lieu, ?personne).

On peut définir une nouvelle classe Occupé comme l'ensemble des lieux contenants des personnes grâce à la réglé suivante Intérieur :

(?lieu) contient (?lieu, ?personne) -> Occupé(?lieu)

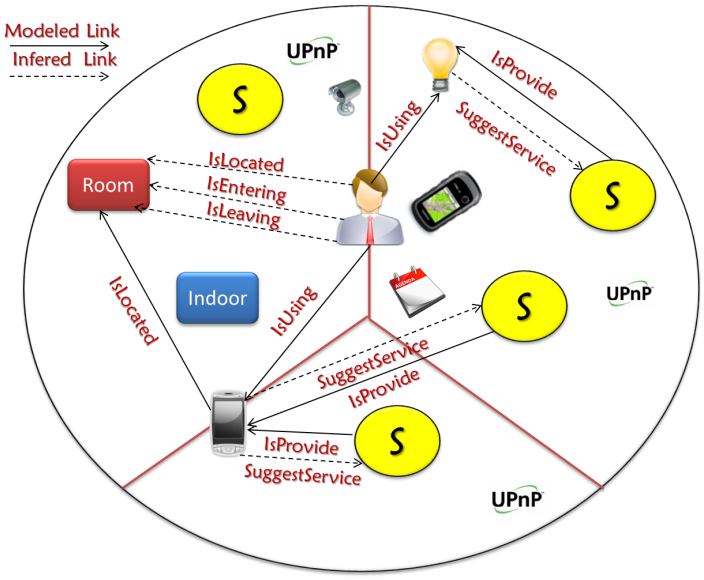

Figure V.7 – Mécanisme de Découverte.

En couplant le contexte de l'utilisateur/Dispositif avec les informations de localisation, il est possible d'aboutir

à la proposition de services implicites pertinents. Par exemple, un utilisateur qui entre dans une pièce se verra proposer des services de luminosité de la pièce où il entre (à travers l'utilisation des propriétés SuggestService, Figure 5.7.

Service(?s), Device(?d), Room(?r), isLocatedIn(?d, ?r), serviceProvidedBy(?s, ?d) == SuggestService(?d, ?s).

8 Conclusion

La modélisation du contexte sous forme d'une ontologie permet de manipuler une représentation formelle de l'environnement. Cette modélisation facilite le partage et la réutilisation des connaissances et permet l'utilisation des mécanismes de raisonnement logique. Nous définissons une ontologie de contexte pour les environnements ubiquitaires composés d'une ontologie supérieure qui modélise les concepts communs à tous les environnements et d'une ontologie de domaine dont les concepts sont propres a un domaine particulier. Nous utilisons les langage OWL qui est un standard de W3C et qui garantit la complétude et la décidabilité des raisonnements sur l'ontologie. Afin de pallier le problème d'expressivité, une solution consiste à étendre le langage OWL avec un langage de règle pour modéliser et résonner dans les environnements ubiquitaires à un haut niveau d'abstraction, le langage SWRL à été utilisé.

Chapitre VI

Notre plate-forme expérimentale : Services proactifs pour Dispositifs

1 Introduction

Dans la perspective de mettre en place notre vision proactive et contextuelle orientée service, nous avons proposé dans le chapitre 4 une architecture générique réunissant les principaux composants d'un systeme pervasif [KB13b]. L'objectif principal est d'avoir des services qui répondent à des besoins dans un contexte donné. Cependant, pour que notre vision des SIP soit développée, il est primordiale de mettre en place un outil offrant le moyen de gérer le dynamisme du contexte, des dispositifs, des utilisateurs et implemantant des mécanismes de selection des services candidats. L'informatique pervasive nous impose de traiter un grand nombre de dispositifs mobiles et hétérogènes et de récupérer les informations contextuelles nécessaires. L'approche orientée service (Service Oriented Computing ou SOC) propose un nouveau moyen pour développer des applications pervasives [PTDL07]. Cette approche utilise des services pour construire des applications plus facilement intégrables et gérables. Le succès du SOC est vraisemblablement lié au faible couplage qu'il favorise. Il permet de développer des applications modulaires et fournir des fonctionnalités sous forme de service.

Dans ce chapitre, nous présentons tout d'abord les éléments nécessaires pour le bon fonctionnement de cette architecture. Ensuite, nous la détaillons en présentant les différents modules qui permettent d'utiliser les services de façon interopérable, décentralisée et proactive.

2 Les architectures orientées services

Les architectures orientées services (SOA) sont un paradigme utilisé pour concevoir des applications distribuées et pouvoir les déployer facilement [PTDL07]. Elles permettent de créer des applications dynamiques à partir d'un ensemble d'entités de base appelées services. Un service est défini comme le moyen de réunir les besoins des consommateurs et les capacités des fournisseurs [RCK10]. Le but principal de l'approche à service est de permettre la définition et l'utilisation de brique logicielle peu dépendante. En réduisant ces dépendances, chaque élément peut évoluer séparément. Ainsi, les applications décrites en termes de services exhibent une plus grande flexibilité. Ce faible couplage entre les composants de l'application provient du motif d'interaction proposé par l'approche à service [FTL+11]. En effet, ce style architectural est basé sur 3 acteurs, Figure 6.1 :

- **Les fournisseurs de services** qui offrent des services.
- **Les consommateurs de services** qui requièrent et utilisent des services offerts par des fournisseurs.
- **Un courtier de service (Registre)** permettant aux fournisseurs de publier leurs services et aux consommateurs de découvrir et de sélectionner les services qu'ils veulent utiliser.

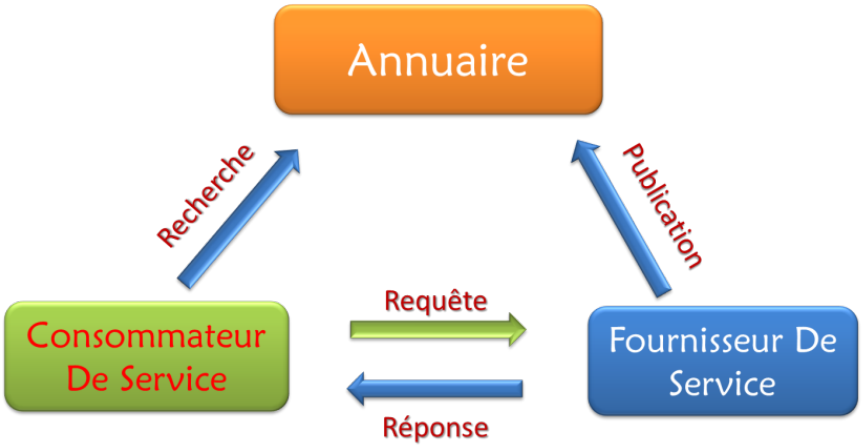

Figure VI.1 – L'approche service.

L'approche à service propose trois primitives d'interaction :

- **La publication :** les fournisseurs de service s'enregistrent auprès du courtier.
- **La découverte :** les consommateurs de service interrogent le courtier afin de trouver les fournisseurs de services qu'ils requièrent.
- **La liaison :** une fois découvert, le consommateur de service peut se lier à un fournisseur de service et utiliser le service qu'il fournit.

L'avènement du Web a fait prospérer un grand nombre d'applications. Un problème principal est apparu : l'hétérogénéité. Les systémes fournissant des services devaient être capables de répondre aux besoins d'un grand nombre de clients hétérogènes. Avec les SOA classiques, le choix d'un langage de programmation, d'un format de représentation des données, ou d'un protocole de communication devait se faire de façon à être compatible avec les consommateurs et fournisseurs du système conçu.

La contribution principale des architectures orientées services Web a été de normaliser les descriptions et les échanges entre les services [BHM+04], [Erl05]. Le standard de description définit le langage WSDL, basé sur XML. Les descriptions contiennent la liste des opérations que fournissent le service et les messages qu'il supporte. D'autres standards permettent de réguler la façon dont interagissent les services et les consommateurs, par exemple SOAP pour les invocations de méthodes et les échanges d'informations. Les technologies du Web ont apporté aux SOA une grande interopérabilité. Les protocoles de communication entre les fournisseurs et les consommateurs reposent sur un réseau de transport TCP/IP et en général sur le protocole de session HTTP. SOAP est une combinaison d'une grammaire XML et du protocole HTTP. Avec les descriptions WSDL et des registres de services UDDI, Figure 6.2.

Les services fournissent des propriétés importantes concernant le découpage de fonctionnalités, et la façon dont elles sont utilisées [FTL+11]. Voici les caractéristiques principales, et les points communs avec les dispositifs d'informatique pervasive qui nous orienterons vers l'utilisation des SOA pour notre infrastructure :

- **Le faible couplage :** les services n'ont pas de dépendance directe entre eux. L'absence d'un service n'empêche

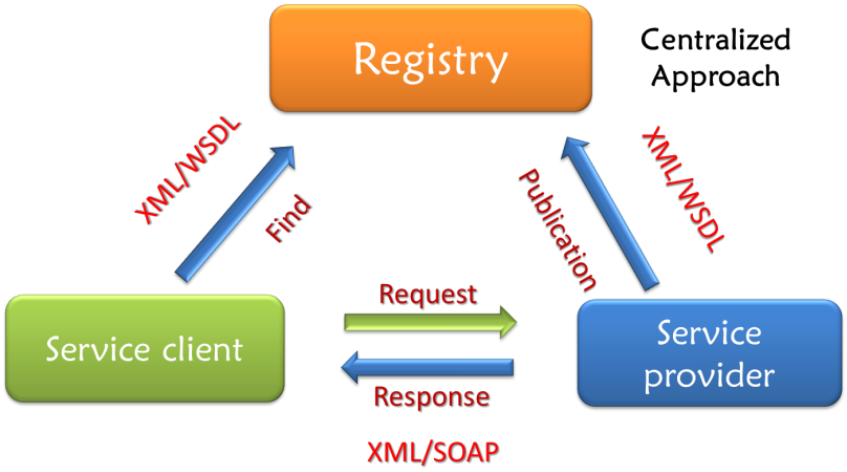

Figure VI.2 – Architectures orientées services.

pas un autre de remplir sa fonctionnalité. Ils sont indépendants. Les dispositifs ont aussi cette propriété.

- **La liaison tardive :** la liaison entre les fournisseurs et les consommateurs a lieu seulement lorsqu'un fournisseur est trouvé et lorsque le consommateur le demande.
- **Le masquage de l'hétérogénéité :** grâce au faible couplage, un consommateur n'a pas à connaitre les détails d'implémentation du fournisseur de service, ni à sa localisation précise.
- **Encapsulation :** toute fonctionnalité peut être encapsulée dans un service et ainsi faire partie de l'ensemble des services disponibles pour créer des applications. Les dispositifs fournissent aussi un ensemble de fonctionnalités ; il est ainsi possible de les encapsuler dans des services au niveau fonctionnel.
- **Contrats et abstraction :** les services décrivent les fonctionnalités qu'ils offrent dans des contrats. Les services doivent se conformer à ces contrats, car c'est le seul moyen pour les autres entités d'obtenir des informations sur leurs fonctionnalités.
- **Découverte dynamique :** les services sont découverts avec certains critères et peuvent être remplacés pendant l'exécution. Les consommateurs de services effectuent une requête au registre pour découvrir les fournisseurs correspondant à leurs critères. Ils obtiennent le contrat du fournisseur, ainsi qu'une référence permettant de le contacter, par exemple une URL.

Toutes ces propriétés communes aux services et aux dispositifs, nous ont poussé à considérer l'utilisation des SOA comme infrastructure pour l'informatique pervasive. Dans ce cas, les dispositifs et les systèmes d'informations sont représentés par des services. Les applications sont des compositions de ces services, en relation constante avec l'infrastructure pour en observer les changements (apparition ou disparition d'un service) et y réagir si besoin, Figure 6.3.

Pour valider l'utilisation des services comme infrastructure d'informatique pervasive, nous devons étudier si les propriétés que nous avons décrites dans l'introduction peuvent être prises en compte : l'hétérogénéité, la proactivité, et les déconnexions fréquentes dues à la mobilité. Les WSOAD (Web Service Oriented Architecture for Devices) ont été conçus précisément dans ce but. Nous allons étudier quelles évolutions ayant apportées depuis les SOA classiques que nous venons de voir.

Figure VI.3 – SOA l'infrastructure pervasive.

3 Évolution des SOA vers les WSOAD

Depuis leur création pour l'informatique distribuée, les SOA ont évolué suivant plusieurs directions : le Web, l'informatique mobile et pervasive. Malgré ces différentes directions, un nouveau type de services regroupant toutes les avancées est né : les architectures orientées services web pour dispositifs (WSOAD), Figure 6.4. Dans cette section, nous allons étudier les diverses évolutions, en montrant leur intérêt particulier pour l'informatique pervasive : l'interopérabilité, amenée grâce aux technologies du Web, la proactivité, fournie par les communications événementielles, et les problèmes dû aux déconnexions fréquentes de l'informatique mobile en compte par la découverte décentralisée [KB14b].

3.1 SOA pour dispositifs

Les capacités de traitement des dispositifs mobiles sont réduites à cause des déconnexions fréquentes, de la taille mémoire et à cause de la batterie. Les programmes doivent ainsi être les plus efficaces et pertinents possible [BGL07]. La mise en place de systèmes à file d'attente ou d'architectures complexes de publication/souscription, utilisant largement ces ressources est rarement envisageable [CJFY05].

Les services pour dispositifs sont des services légers, étendus par deux points fondamentaux : les communications par événements, et la découverte décentralisée. La découverte décentralisée permet de découvrir sans se baser sur une entité statique. Les communications par événements permettent aux applications basées sur des dispositifs d'utiliser de façon efficace les interruptions matérielles et apportent la proactivité nécessaire aux interactions entre les dispositifs prenant part dans les applications d'informatique pervasive. Elles ajoutent de plus une dynamicité dans ces interactions, en permettant à tout moment aux entités concernées de les reconfigurer.

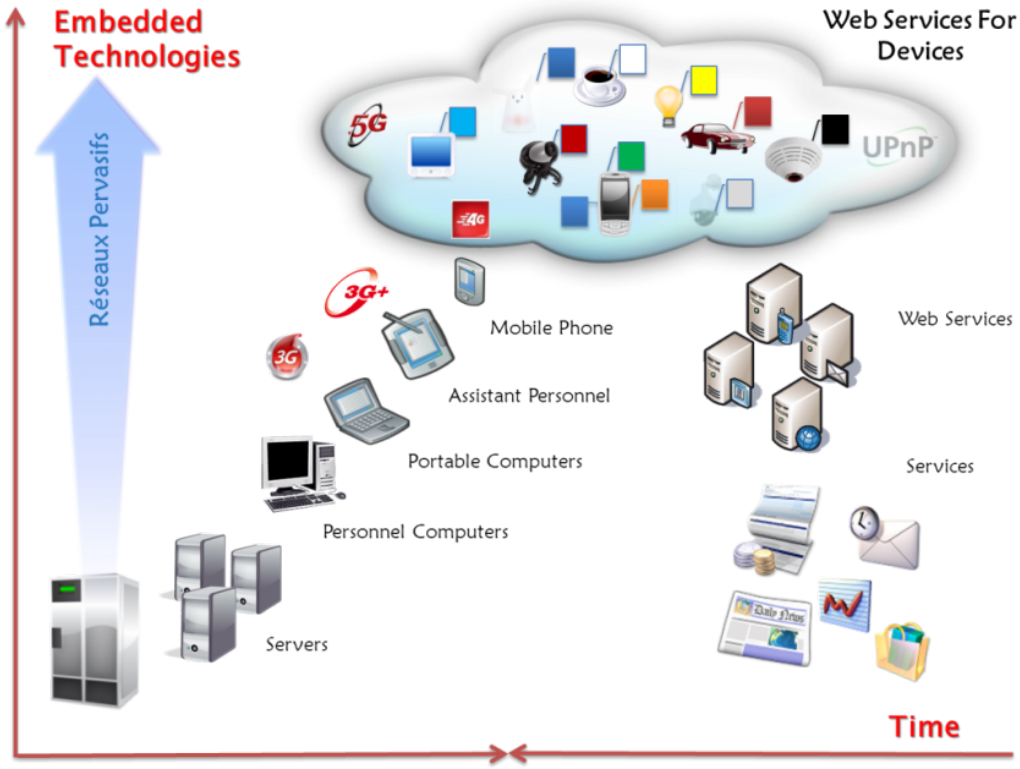

Figure VI.4 – Des Services Vers des Services pour Dispositifs.

3.2 WSOAD : Web Service Oriented Architecture for Device

L'utilisation des technologies Web amène une grande interopérabilité entre les services [FLRT08], la découverte décentralisée permet de découvrir les services dans des environnements mobiles, et les communications par événements ajoutent la proactivité aux interactions entre fournisseurs et consommateurs et les découplent encore plus. Ces deux apport ont contribués largement à l'évolutions principales des SOA classiques vers les WSOAD [ws414], Figure 6.4.

3.2.1 Découverte dynamique décentralisée

L'autre évolution des SOA apportée par les SOAD est la découverte. La mobilité des utilisateurs, et des dispositifs de leurs réseaux personnels, provoque de fréquentes déconnexions et changements de réseaux. Les entités qui seront présentes pour créer une application ne sont donc pas connues, et doivent être découvertes de façon dynamique.

Dans les SOA, nous avons vu qu'un registre de services était utilisé pour maintenir une liste de fournisseurs de services disponibles. Ce registre est utilisé par les consommateurs pour découvrir les fournisseurs. Un prérequis est donc que les consommateurs aient une référence vers un registre pour pouvoir construire l'application lors de l'exécution. Dans les réseaux mobiles, on ne peut pas connaître à l'avance l'adresse des registres, ni même supposer qu'il en existera un. Des mécanismes de recherche décentralisée ont alors fait leur apparition, dans des standards industriels (SLP, Jini, Bluetooth SDP, Salutation, Bonjour). Ils permettent aux consommateurs de découvrir les fournisseurs sans passer par un registre centralisé. Le mécanisme de recherche est en fait ramené dans les fournisseurs et les consommateurs qui communiquent donc directement entre eux. La phase de découverte utilise alors la diffusion de

messages.

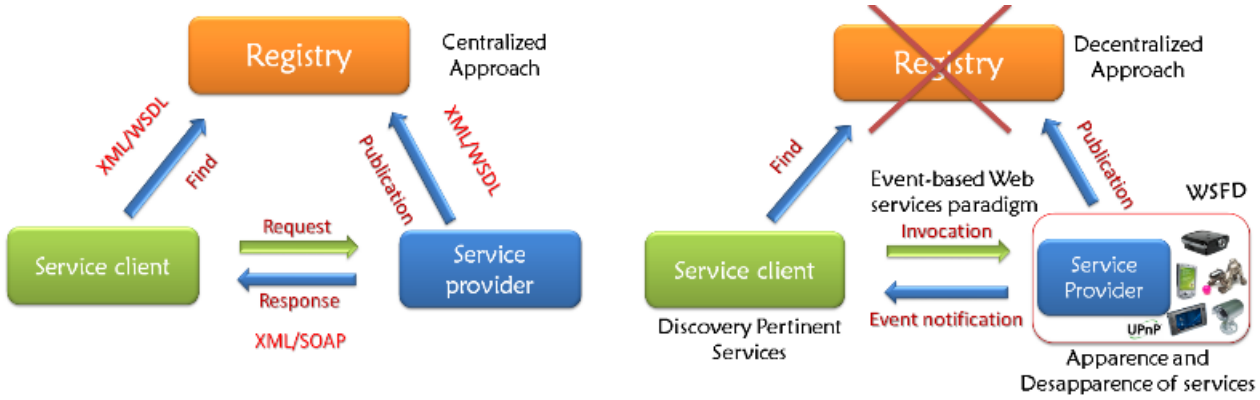

Figure VI.5 – Web Service Oriented Architecture vers Web Service Oriented Architecture pour dispositifs.

Avec la découverte décentralisée ou proactive, Les fournisseurs de services émettent des annonces par diffusion à tout le réseau lorsqu'ils sont lancés, ou qu'ils sont en train de quitter : cette découverte par annonce est asynchrone, et fournit la propriété de proactivité au mécanisme de découverte. Lorsque les consommateurs ont obtenu une référence vers les fournisseurs, en général une URL, ils effectuent une requête pour obtenir le contrat du service afin de pouvoir interagir avec eux au niveau fonctionnel. La découverte est la première étape de la création d'applications à partir d'une infrastructure de services, Figure 6.5. En informatique Pervasive, c'est d'autant plus le cas parce que les environnements dans lesquelles elles seront créées ne sont pas connus à l'avance. La découverte dynamique et décentralisée permet de découvrir les services sans les connaître à priori, et sans se baser sur une entité statique.

3.2.2 Communications par événements

Le modèle événementiel introduit un découplage total entre les services serveurs et les services clients [FTL+11]. Les communications sont basées sur des échanges d'événements qui sont des transitions ponctuelles et significatives d'états de l'environnement physique. Le modèle événementiel est plus adapté à la gestion des environnements ubiquitaires, Figure 6.6.

Les communications par événements apportent la proactivité nécessaire aux interactions entre les dispositifs prenant part dans les applications d'informatique pervasive. Elles ajoutent de plus une dynamicité dans ces interactions, en permettant à tout moment aux entités concernées de les reconfigurer.

Les WSOAD (Web Service Oriented Architecture for Device) naissent du regroupement de ces trois évolutions, auxquelles est ajoutée l'interopérabilité des WSOA [FTL+11]. Les services de ces architectures doivent donc être connectés à des réseaux IP. La tendance actuelle et celle prévue pour le futur exposent que de plus en plus de dispositifs seront reliés à un réseau Ethernet ou Wi-Fi, donc utilisant le protocole IP. Pour les dispositifs qui utilisent des protocoles propriétaires ou de plus faible portée, des programmes faisant le pont entre ces protocoles et les services Web pour dispositifs seront conçus. Nous les appelons les bridges. Deux technologies existent dans cette famille d'architectures : UPnP et DPWS. Les deux sont nées d'une impulsion de Microsoft en tant que protocoles de découverte et de communication avec des dispositifs mobiles comme des lecteurs multimédias ou des imprimantes.

UPnP : l'Universel Plug'n Play est né en 1999 dans le but d'étendre le Plug n Play, qui permet de découvrir les composants et les périphériques d'un ordinateur, aux dispositifs d'un réseau [UPn00]. UPnP décrit les interactions

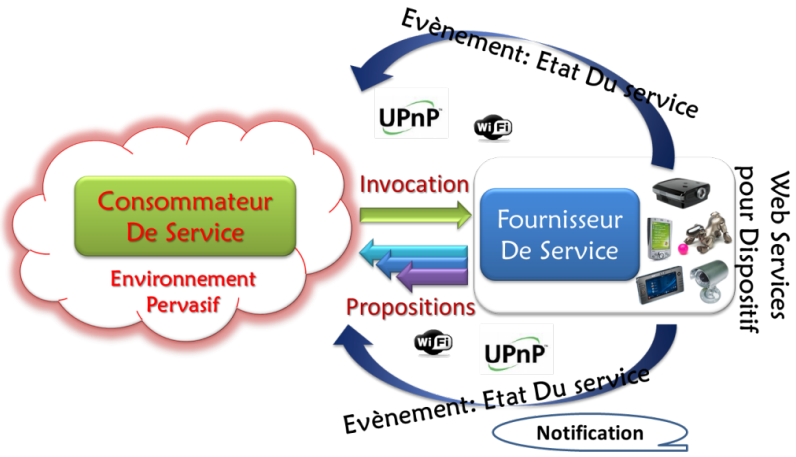

Figure VI.6 – communications par événements.

entre les Devices UPnP (fournisseurs) et les points de contrôle (control point, les consommateurs) avec cinq protocoles : l'adressage, la découverte, la présentation, l'invocation, la notification d'événements. Le format de description des services (contrats) est aussi défini .

Un Device UPnP représente le matériel global fournissant les fonctionnalités, et les moyens d'interaction avec l'infrastructure [UPn00]. Il se décompose en services, qui sont les interfaces réelles des fournisseurs de services. La hiérarchie est donc la suivante (Figure. 6.7) :

Un Device appelé root contient un ou plusieurs services [UPn00]. Les services contiennent une liste d'actions et de variables d'état. Les actions sont les méthodes qui pourront être invoquées dans le service. Les variables d'état ont une sémantique complexe. Chaque argument des actions doit être couplé à une variable d'état pour définir son type. Ce sont les variables d'état passives. D'autres variables d'état, les actives, sont celles qui définissent les événements d'un service. un Device complet peut être embarqué dans le Root Device. Il est alors appelé Embedded Device.

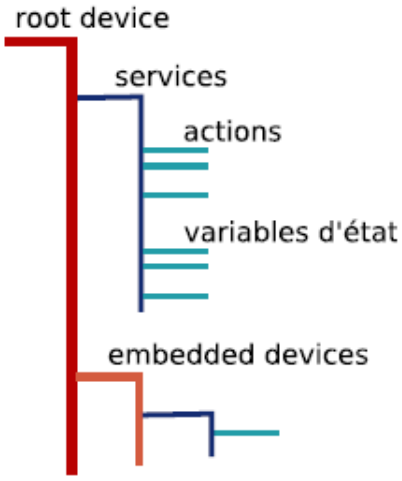

Figure VI.7 – UPnP.

Les devices et les services ont chacun un fichier de description, un contrat. Celui du device contient le type et

le nom du Device, sa liste de services, l'URL de présentation, et d'autres informations non-fonctionnelles comme le fabriquant, le modèle ou le numéro de série. Pour chaque service sont indiquées une URL vers le fichier de description du service, une pour l'invocation d'actions et une pour la souscription aux variables d'état du service en question. Le fichier de description d'un service est son contrat et contient une liste d'actions et de variables d'état. Chaque méthode est liée à des variables d'état pour les arguments, dont ceux de retour, optionnels. Les descriptions des variables d'états définissent leur type, et éventuellement une liste de valeurs ou une plage de valeurs qu'elles peuvent prendre.

Le protocole d'adressage utilise DHCP ou Auto-IP , standardisés par l'IETF. Il définit de quelle façon un device UPnP doit obtenir ou choisir une adresse IP lorsqu'il n'a pas de moyen d'en obtenir une. La découverte utilise un protocole HTTP modifié pour être utilisé sur UDP au lieu de TCP, appelé SSDP (Simple Service Discovery Protocol). Les deux modes de découverte sont pris en charge (par annonce et par recherche). L'utilisation d'un registre dynamique n'est pas prévu par la technologie UPnP. Les messages sont envoyés par diffusion sur UDP. Les messages point à point utilisent aussi UDP. Les requêtes de récupération des descriptions utilisent HTTP sur TCP. Le protocole de présentation est simplement HTTP, et est utilisé dans UPnP pour permettre aux dispositifs de fournir une pageWeb d'administration ou de résumé de leur état. Les invocations d'actions sont permises par le standard SOAP, dans une version simplifiée. Les communications événementielles utilisent un protocole qui est, une fois de plus, propre à UPnP appelé GENA (General Event Notification Architecture). Il est utilisé à la fois pour la découverte par annonce et pour la publication, la souscription et la notification des variables d'état actives comme sources événements. La souscription ne peut se faire qu'avec la granularité d'un service, et concerne toutes les variables d'état actives que le service contient. C'est une limitation importante d'UPnP. GENA utilise un mécanisme de leasing pour la découverte et la souscription aux événements, Figure 6.8.

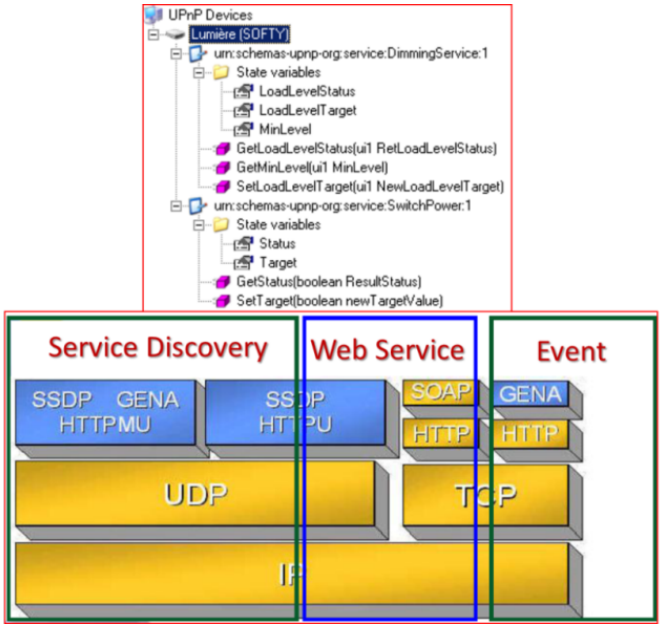

Figure VI.8 – Architecture d'un Device UPnP.

DPWS : le Device Profile for Web Services reprend un ensemble de spécifications des services Web et y ajoute de nouvelles extensions pour la découverte et les communications événementielles [DM09]. DPWS est né en 2004,

comme évolution d'UPnP, il est parfois appelé UPnP v2. Cependant, ils n'ont en commun que les concepts et ne sont pas interopérables.

Les descriptions des services DPWS utilisent le format standard des services Web [DM09], WSDL. Les fonctionnalités des services Web sont toujours disponibles, l'invocation de méthode est donc réalisée en utilisant le standard SOAP. Le protocole de sécurité WS-Security est utilisé. La spécification WS-Discovery est utilisée pour la découverte. Les communications événementielles sont apportées avec la spécification WS-Eventing. Cet ensemble de protocoles permet aux services DPWS d'être interopérables avec des services Web classiques. Contrairement à UPnP, DPWS ne définit pas de profils standardisés pour des types de dispositifs. Grâce à sa couche commune avec les services Web, DPWS cible l'intégration de services fournis par des dispositifs dans des applications d'entreprise. Cependant, il convient très bien à la création d'applications mobiles.

Aujourd'hui, UPnP [UPn00] est très utilisé dans des produits industriels comme les routeurs et les passerelles domestiques, ainsi que dans les dispositifs de partage de données et de rendu multimédia. D'autres dispositifs intègrent parfois un Device UPnP, mais ils n'utilisent que les protocoles de découverte et de présentation. Ils permettent alors de donner un lien à l'utilisateur vers la page Web de téléchargement d'un pilote pour l'installer correctement. Ce genre d'utilisation est proche du non-sens puisqu'UPnP définit tous les protocoles nécessaires pour interagir avec les dispositifs sans avoir besoin de pilotes et en étant interopérable avec les systèmes existants. Dans le cadre de cette thèse la pile UPnP a été choisie.

3.3 Récapitulatif

Les services Web pour dispositifs offrent les propriétés nécessaires à l'infrastructure d'informatique Pervasive : interopérabilité, dynamicité, réactivité. Les dispositifs peuvent être découverts, sans être connus au moment de la conception des applications. Cette infrastructure logicielle représente la base des applications opportunistes et permet de collecter les informations contextuelles nécessaires à la sensibilité au contexte. Cependant, nous devons mettre en place des techniques d'orchestration ou de composition de services dynamiques pour prendre en compte les variations de l'infrastructure.

4 Introduction de la proactivité dans les environnements pervasifs

Le terme informatique proactive a été introduit par Tennenhouse [Ten00]. Il décrit l'évolution de l'informatique interactive. L'informatique proactive utilise des systèmes qui se connectent au monde physique et nécessitent que peu d'intervention humaine. Dans [VVV08], le système anticipe le désir de l'utilisateur et ajuste automatiquement l'éclairage de l'environnement. Au cours des dernières années, il ya eu un nombre croissant de travaux avec des caractéristiques dynamiques. Dans le domaine des systemes proactifs, les services sont choisis en fonction du contexte courant de l'utilisateur (localisation, préférences). Les préférences de l'utilisateur ne sont généralement pas spécifiés explicitement par les utilisateurs eux-mêmes, mais plutôt motivées sur la base du comportement de l'utilisateur précédent, la suite, la liste des services sélectionnés et proposés de façon proactive à l'utilisateur sans une demande explicite de sa part.

Dans le projet DYNAMOS [PLZ04], le système informe sur des situations critiques. Il offre des services proactifs basés sur la description du contexte de ses utilisateurs. [PvBMH09] présente un système de surveillance des patients répartis géographiquement qui utilise des techniques de sélection de services proactifs pour leur service d'intervention d'urgence. Un autre exemple sur la santé omniprésente [FFL11], il vise à fournir des services personnalisés en fonction

de l'état de santé du patient, qui est acquis à partir d'un réseau de capteurs sans fils. Les systèmes réactifs attendent un ordre explicit de l'utilisateur pour lancer une action et les systèmes proactifs essaient de dériver une action à la base des informations contextuelles (localisation, préférences utilisateurs etc.), Figure 6.9.

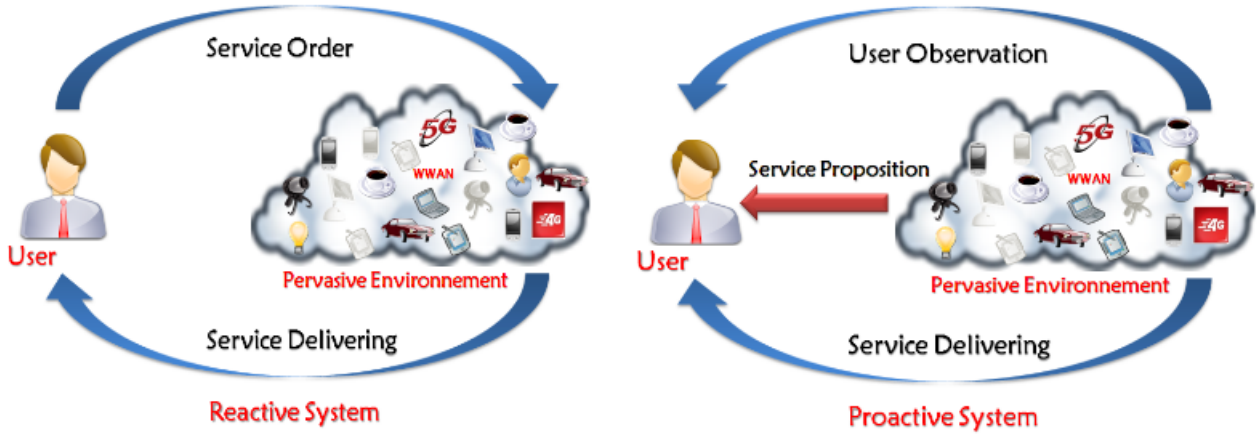

Figure VI.9 – Mécanisme de proactivité.

5 Aperçu de la solution

Après notre revue de la littérature dans le Chapitre 2 et le Chapitre 3, nous avons constaté que l'orientation service, la sensibilité au contexte, ainsi que l'approche proactive représentent des approches très prometteuses à prendre en considération afin de concevoir un SIP transparent.

De grands efforts de recherche ont été menés sur le sujet de la découverte des services, [RHC+02] et [BMPG+07]. En plus des approches citées précédemment, une autre vision est proposée par des travaux tels que [BCMT06], [CSKL13], [NKS12] et [Kwo12]. Ces approches proactives, prennent l'importance des besoins de l'utilisateur lors du choix des services.

Nous allons nous concentrer essentiellement sur les challenges : la selection et la proaction de services. Nous nous intéressons ainsi aux différentes approches mises en place par les systèmes à base de services. En d'autres termes, la demande explicite et la découverte des services ont prédominé les environnements pervasifs.

5.1 Notre vision :couplage entre Services, Contexte et Proaction

Notre vision des Systèmes d'Information Pervasifs est basée sur les notions de Proaction, de contexte et de services [KB14b]. Cette solution permet de gérer l'hétérogénéité et la dynamicitée de l'environnement pervasif à travers une approche contextuelle.

1. Notre vision se caractérise par son orientation service, cela va permettre de répondre au besoin de gestion de l'hétérogénéité et la dynamicité qui caractérisent l'environnement pervasif dans lequel évoluent les SIP et des actions que le système propose afin de satisfaire les besoins des consommateurs.
2. Notre vision se base sur une orientation contexte. Ceci va permettre de proposer des services pertinents sensibles au contexte (l'utilisateur et l'environnement). Nous avons soulevé le rôle central que joue la notion de contexte

dans les systèmes sensibles au contexte (Chapitre 2). Notre approche présente les SIP comme des systèmes sensibles au contexte, qui s'adapte aux changements de l'environnement.

3. Enfin, et dans la perspective de répondre au mieux aux exigences des utilisateurs, notre vision des SIP est orientée Proaction. Ainsi, les SIP prennent en considération la localisation pour, d'une part, capter le contexte des utilisateurs, et d'autre part, répondre d'une façon plus appropriée à leurs besoins.

Nous considérons que la satisfaction de l'utilisateur dans un SIP dépend du contexte dans lequel se trouve en lui proposant des services d'une façon implicite (Le Dispositif ou l'utilisateur n'exprime pas de requête explicite). Pour nous, le contexte impacte directement le choix des services qui seront exécutés.

5.2 Les modules de l'architecture

Pour le bon fonctionnement de notre architecture, certains prérequis doivent être établis au préalable. En effet, et comme nous l'avons décrit dans le Chapitre 5 [KB14b], pour pouvoir gérer le contexte de l'utilisateur, il faut avoir décrit ces différents éléments. Ceci nécessite la description au préalable d'une ontologie multi-niveaux de contexte qui va décrire tous les concepts de contexte qui sont jugés pertinents dans les espaces de services déterminés. Cette ontologie multi-niveaux de contexte est exploitée, par la suite, par le gestionnaire de contexte, afin d'aider à la capture et à la description de contexte courant de l'utilisateur, mais aussi des services et des capteurs.

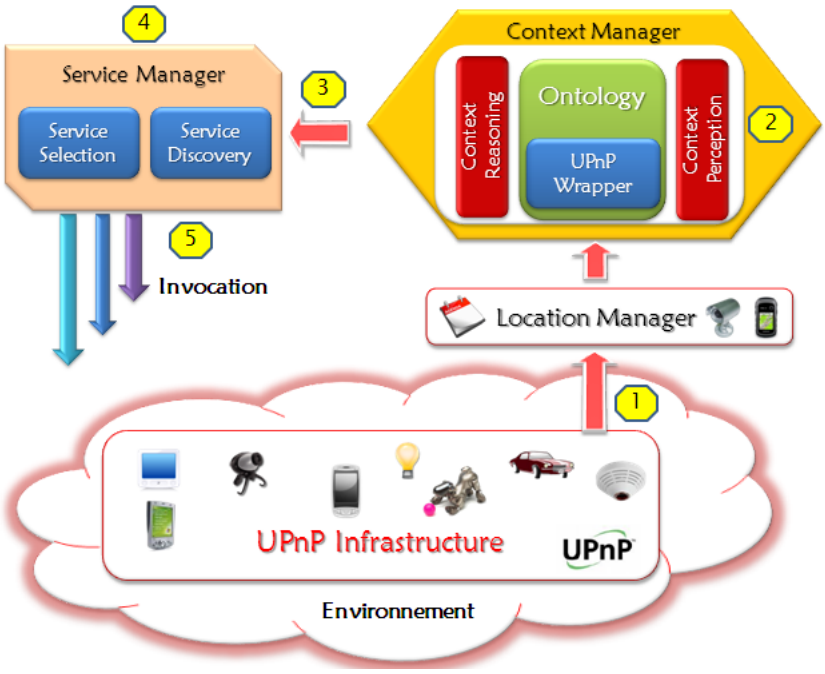

Figure VI.10 – Architecture du framework.

De plus, cette architecture offre des modules de découverte et de prédiction de services qui permettent de sélectionner et de prédire le service qui répond au mieux à l'utilisateur dans un contexte donné. Nous présentons dans cette section, l'architecture composée de trois modules principaux, Figure 6.10 [KB14b] :

– **Module de gestion de localisation :** responsable de la localisation des utilisateurs et dispositifs afin de leurs proposés des services proactifs.

- **Module de gestion de contexte :** responsable de la gestion des informations contextuelles capturées de l'utilisateur et des entités qui composent l'espace de services également (services et capteurs) ;
- **Module de gestion de services :** responsable de la découverte et de la sélection du service qui répond au mieux au besoin immédiat de l'utilisateur dans son contexte courant et responsable de la prédiction des services les plus appropriés qui peuvent répondre à un besoin futur de l'utilisateur.

La Figure 6.11, présente le diagramme de séquence de notre prototype.

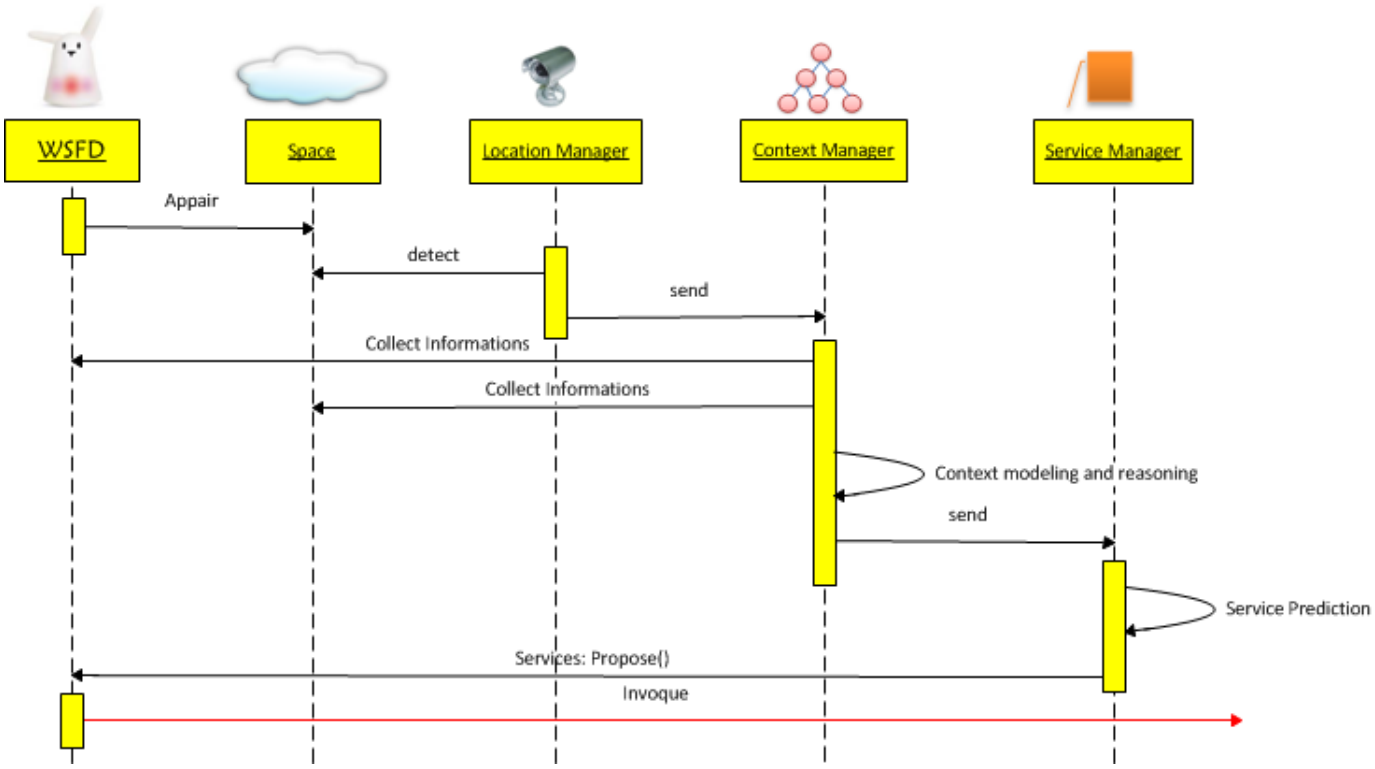

Figure VI.11 – diagramme de séquence de notre prototype.

5.2.1 Gestionnaire de localisation

La localisation est une information de contexte capitale, sinon la plus importante, à partir de laquelle des services peuvent être proposés et inférées. Cette information est généralement déduite de données issues de capteurs (e.g. détecteurs de mouvements, contacteurs), Figure 6.12. Dans nos travaux, nous considérons les activations de services de l'utilisateur/Dispositif est associés à leur localisation [KB14b]. À partir de cette représentation logique, des règles sont exprimées pour inférer des propriétés comme isLocatedIn, isEntering, et isLeaving suivant l'activation de services. Considérons par exemple la règle suivante :

isLocatedIn == LocalDevice(?d) and Room(?r) and User(?u) and isLocatedIn(?d, ?r) and isUsing (?u, ?d) == isLocatedIn(?u, ?r)

Un utilisateur U qui utilise un Dispositif locale D qui se trouve dans une pièce R, se trouve lui-même dans cette pièce.

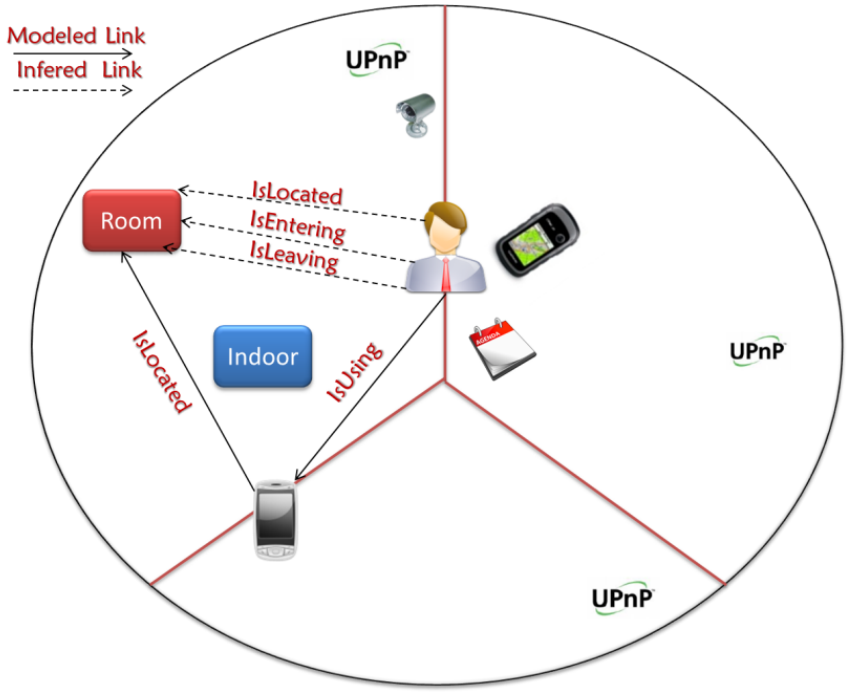

Figure VI.12 – Mécanisme de Localisation.

5.2.2 Gestionnaire de contexte

Le module de gestion de contexte est l'élément de l'architecture responsable de la gestion des informations contextuelles de l'utilisateur (dispositifs), Figure 6.13. Autrement dit, il se charge de collecter et de modéliser les informations contextuelles courantes des utilisateurs (dispositifs) qui interagissent dans le SIP. Le rôle principal de ce module est de cacher la complexité de la gestion de contexte en proposant une manière uniforme d'accéder aux informations contextuelles.

1. L'acquisition des informations contextuelles.
2. Le traitement et la modélisation de ces informations.
3. L'interprétation.
4. Le stockage.

En s'inspirant de ces travaux [DAS01], notre module de gestion de contexte est conçu pour recevoir, du composant responsable de l'acquisition de contexte à partir des capteurs physiques et logiques, les informations contextuelles brutes relatives à chaque utilisateur. Ces informations sont par la suite modélisées et interprétées en respectant le modèle de contexte représenté par l'ontologie multi-niveaux de contexte [KB14b].

– **Capture du contexte :** La fonction principale assurée par cette couche est la capture du contexte. Le contexte peut parvenir de plusieurs sources comme les capteurs physiques, les capteurs logiques ou les capteurs virtuels

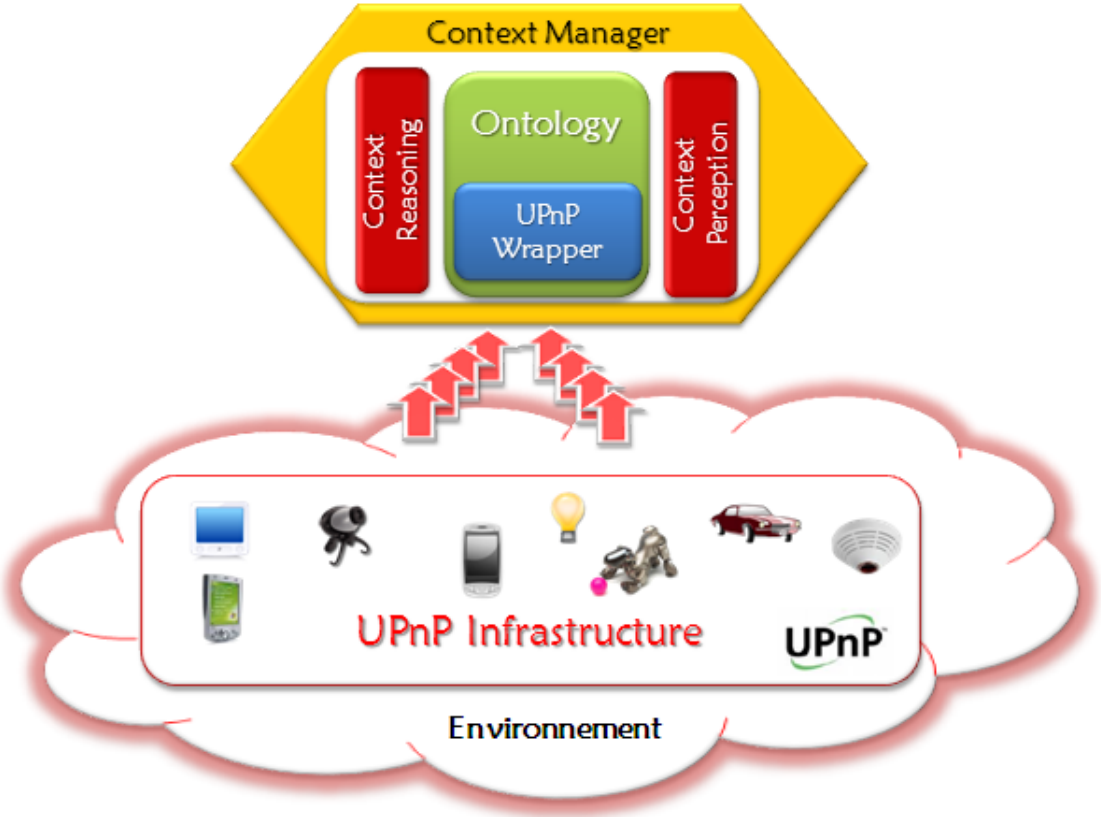

Figure VI.13 – Gestionnaire de contexte.

comme présenté dans l'état de l'art de ce mémoire (chapitre 2). Ensuite, les données contextuelles capturées sont stockées dans un registre contextuel sous le format XML.

– **Conception de l'ontologie du contexte :** Dans le chapitre 2, nous avons justifié le choix de l'ontologie pour modéliser le contexte. Les ontologies de haut niveau (Upper ontologies) sont celles qui sont le plus souvent utilisées puisqu'elles offrent des concepts universels, indépendants du domaine et qui sont flexibles à une adaptation pour différentes applications qui ne réalisent pas forcément les mêmes traitements et n'appartiennent pas au même domaine. Nous étendrons cette ontologie avec les concepts du protocole UPnP (service et device). Cette approche est représentée dans la Figure 6.14. Nous partons alors de l'ontologie de haut niveau comme point de départ qui represente les concepts fondamentaux (Space, User Preferences, Network), Ces concepts sont reliés entre eux par des relations. On y ajoute les concepts du protocol UPnP afin d'obtenir notre modèle de contexte [KB14b]. Ce modèle permet la déduction du contexte au moyen des règles d'inférences, Figure 6.15. Basé sur la sémantique formelle des logiques de description, l'ontologie permet de représenter les concepts ainsi que la nature des liens entre ces concepts. OWL (Ontology Web Language) est le langage utilisé pour représenter notre ontologie. La définition de ces règles nécessite le choix d'un langage de règles, d'un outil d'édition des règles et d'un raisonneur. Pour définir ces règles, nous avons utilisé le langage SWRL. Une combinaison du langage OWL avec celui du Rule Markup Language.
  Les concepts généraux de l'ontologie contiennent des sous entités ou sous classes qui présentent des extensions de plus haute spécificité et facilitent l'intégration des concepts propres au domaine. OWL permet la structuration

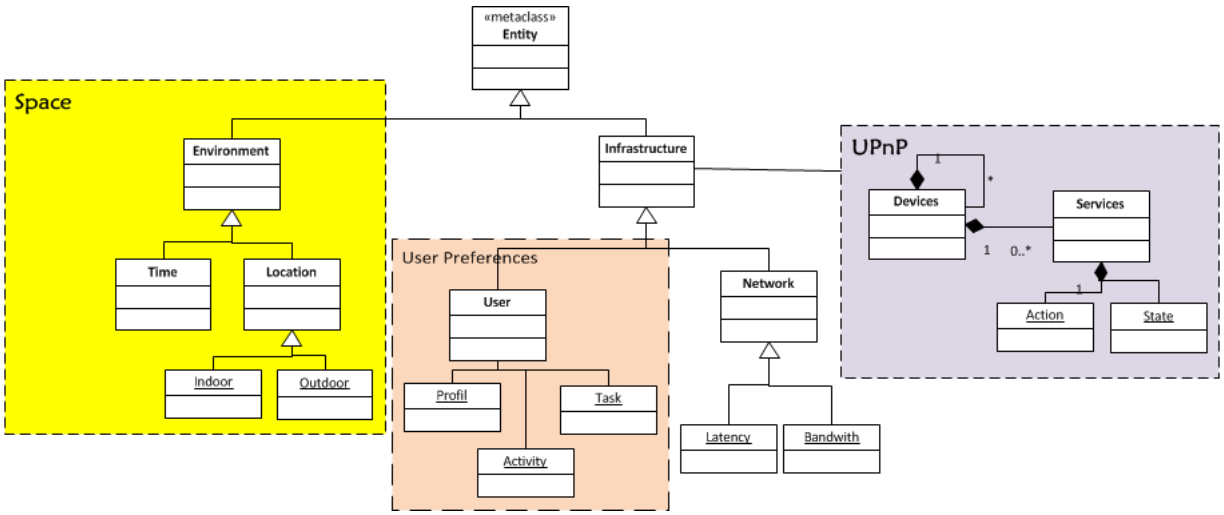

Figure VI.14 – Ontologie du gestionnaire du contexte

des éléments de manière hiérarchique par la notion subClasseOf Ainsi, la localisation est divisée en espace intérieur (Indoor) et extérieur (Outdoor). Exemple, lorsque le nombre de personnes présentes dans une pièce est supérieure à trois et que l'application Powerpoint est active, le système déduit que l'activité de cette pièce est une présentation : People (Room2401 ; >= ; 3) and Application (PowerPoint ; Running)) RoomActivity (2401 ; Présentation). Notre modèle sépare en deux parties la modélisation du contexte : la première s'intéresse aux concepts décrivant le système à un haut niveau d'abstraction alors que la deuxième reprend les concepts spécifiques à un domaine. Le concept générique "Indoor" se décline par exemple en concepts "Building", "Room", dans le domaine de la Domotique.

Prenons l'exemple de la propriété isLocatedIn, définie comme transitive, Figure 6.15. Le moteur d'inférences, sachant que Amine, isLocatedIn, Bedroom et que Bedroom, isLocatedIn, House, peut inférer que Amine, isLocatedIn, House. Les inférences logiques réalisées par un moteur d'inférences permettent de déduire des connaissances implicites à partir de l'ontologie (modèle explicite) et de règles s'y appliquant.

- **Résonner sur les ontologies :** Le rôle du gestionnaire de contexte est d'identifier la situation courante pour l'ensemble des contextes, Figure 6.16. Pour ce faire, il interroge la base de connaissances de manière à vérifier les différents prédicats issus de la modélisation des différents contextes. Une fois qu'une situation d'un contexte donné est identifiée, le Gestionnaire de Contexte transmet au gestionnaire services le context courant de la situation. Les environnements pervasifs sont dynamiques par nature. Les dispositifs entrent et sortent de l'environnement ce qui entraine une disponibilité dynamique des services, Lorsqu'un service annonce sa présence a l'infrastructure par le biais d'un protocole de découverts de service (UPnP dans notre cas). L'infrastructure intègre le nouveau service dans le modèle de contexte. Le modèle de contexte doit refléter cet état et en particulier la disponibilité des services pertinents.
- **La base de connaissances :** La base de connaissances est à l'écoute de l'environnement. Elle est avertie de l'ensemble des évènements issus de l'infrastructure logicielle comme, par exemple, l'apparition ou la disparition de services UPnP. Le Gestionnaire de Contexte n'évalue pas en permanence l'ensemble des prédicats des contextes qu'il gère. L'évaluation des prédicats est en fait conditionnée à la réception d'un ou plusieurs évènements en provenance de la base de connaissances. On parlera des évènements en provenance de l'infrastructure

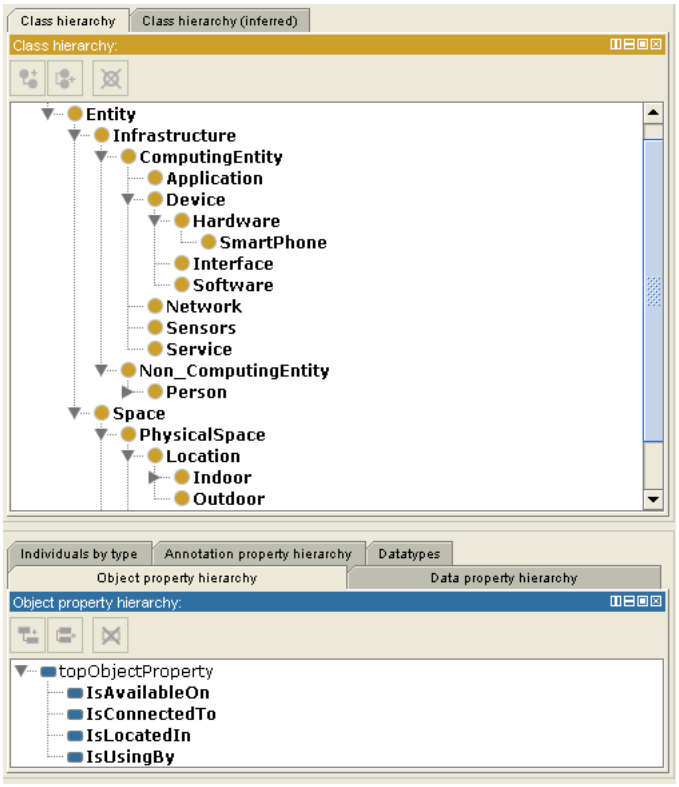

Figure VI.15 – Concepts et relations de l'Ontologie du gestionnaire du contexte

logicielle UPnP issues de :

1. La base de connaissances ne fait que transférer les évènements (la dynamique de l'environnement) qu'elle reçoit au Gestionnaire de Contexte.
2. Traitement du moteur d'inférences qui permet d'inférer de nouvelles connaissances.

– **Le moteur d'inference :** Le moteur d'inference applique des règles logiques aux faits (connaissances) contenu dans la base de connaissances. Cette base contient le modèle de contexte. Le moteur d'inference accepte en entré un ensemble de règle et de faits prévenant de la base de connaissance. Le mécanisme de résonnement permet au moteur d'inférer de nouveaux faits à partir des faits qui sont stockés dans la base. les nouveaux faits sont à leur tour utilisés pour exécuter de nouvelles règles de plus haut niveaux, prenant l'exemple suivant : sonore (?lieu, ?niveau>70 db) and IsLocatedIn(?lieu, ?visiteur). On peut définir une nouvelle classe conférence comme l'ensemble des lieux contenants des visiteurs grâce à la réglé suivante :

sonore (?lieu, ?niveau>70 db) and IsLocatedIn(?lieu, ?visiteur) == Conference(?lieu)

5.2.3 Gestionnaire de Service

Aujourd'hui, nous constatons que l'utilisateur interagit avec une panoplie de dispositifs et de services offerts par l'ensemble des SI qui nous entourent. De grands efforts ont été effectués sur l'adaptation au contexte, surtout à la

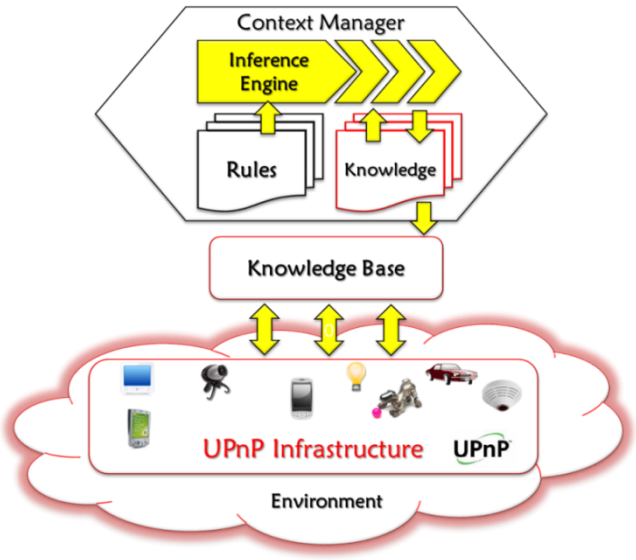

Figure VI.16 – Raisonneur de Contexte.

localisation et aux dispositifs utilisés. Nous proposons, un processus contextuel et proactif fournissant à l'utilisateur le service pertinent parmi les services candidats qui satisfait ses besoins d'une manière implicite et proactive, sans qu'il soit forcé de comprendre des détails sur l'implémentation ou sur les contraintes des dispositifs utilisés, Figure 6.17. Ce processus se base sur un principe de mise en correspondance entre le contexte courant de l'utilisateur et les conditions contextuelles associées aux services, d'autre part. D'une manière générale, l'algorithme de sélection du service candidat que nous proposons se décompose en trois étapes [KB14b].

– **Module de découverte de services :** Un utilisateur interagit dans un espace de services. Ceci consiste à lui offrir en se basant sur sa localisation, le service le plus adapté à son contexte courant et le plus approprié à ses intentions.Par conséquence, l'infrastructure UPnP utilisé dans le cadre de notre thèse, spécifie un ensemble de protocoles basés sur les protocoles standards d'Internet pour mettre en place une architecture à service dynamique. Grâce à ces protocoles, un dispositif compatible UPnP peut dynamiquement rejoindre un réseau, obtenir une adresse IP, annoncer son nom, fournir la liste de ses capacités (services), détecter les autres dispositifs ainsi que découvrir leurs services. Un concept clé dans UPnP est le concept de dispositif (Device). Lorsqu'un dispositif rejoint un réseau, il s'annonce auprès de tous les autres acteurs présents sur le réseau. Cette annonce est diffusée sur le réseau local. Lorsqu'un dispositif quitte le réseau, il émet également un message diffusé sur le réseau pour notifier son départ. Le protocole de découverte proposé par UPnP est Simple Service Discovery Protocol (SSDP). Un point important est qu'UPnP ne possède pas d'annuaire de services.
  Une fois qu'un dispositif est découvert, il est possible d'interagir avec lui afin de lui demander des informations plus spécifiques telles que les services hébergés. Les dispositifs publient leurs variables d'état pour chaque service et décrivent la liste des actions appelables. Ensuite, il est possible d'interagir avec le dispositif via les services proposés. UPnP propose un moyen afin d'être notifié des changements d'état d'un dispositif. Lorsqu'une variable d'état est modifiée, le dispositif émet une notification sur le réseau contenant l'identifiant du dispositif, la variable d'état et la nouvelle valeur. UPnP utilise le protocole General Event Notification Architecture.
  Ce processus représente les services disponibles aux besoins immédiats de l'utilisateur dans un contexte donné. En utilisant la description de services fournit par l'ontologie, les services sont classées en fonction de leurs

informations contextuelles. On sélectionne celui qui répond au mieux à l'utilisateur ou au device dans son contexte courant.

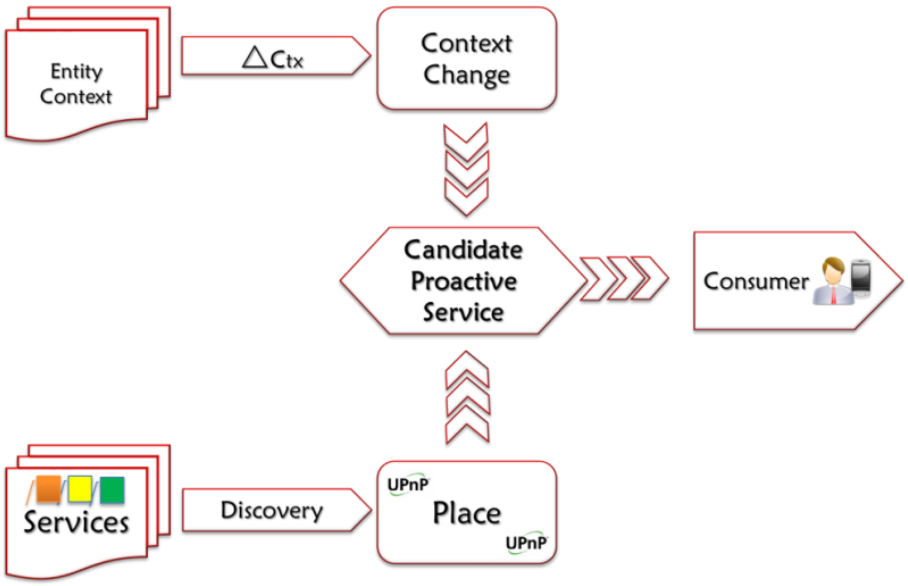

Figure VI.17 – Election du Service Condidat .

- **Module de sélection de services candidat :**
  - **Etape Une :** on récupère le contexte courant de l'utilisateur dans lequel le service va s'exécuter grâce à l'ontologie définie dans le chapitre 5.
  - **Etape deux :** Grace au protocole UPnP, on liste les services présents et candidats à l'invocation.
  - **Etape trois :** pour chaque service disponible, le contexte courant de l'utilisateur est comparé, sémantiquement en se basant sur des mesures de similarités, aux conditions de contexte associées au service. Le score du degré de mise en correspondance entre le contexte courant de l'utilisateur/Dispositif et chacun des services disponibles dans l'environnement est calculé. Ce score va définir le classement. Le service qui obtient le score le plus élevé est proposé à l'utilisateur.

    Notre algorithme de sélection de services candidats, Figure 6.18 prend en entrée, une description du contexte courant de l'utilisateur (Ce) et un ensemble de services disponibles (Sd) dans l'environnement grâce à l'infrastructure UPnP, ajoutant à cela une ontologie multi-niveaux de contexte. L'algorithme de mise en correspondance produit à la sortie, une liste qui contient les couples <CPS, Sscore>. Le service CPS représente le service candidat à l'invocation. Le degré de similarité Sscore représente le degré de similarité entre le contexte du service Sdi et le contexte de l'utilisateur/Dispositif Cei.
    L'algorithme de sélection du service proactive, commence par sélectionner un service (Sdi) parmi les services disponibles dans l'infrastructure UPnP, . Ensuite, le contexte (Cc) courant qui récupére les derniers changements de l'environnement et des preferences de l'utilisateur. Après avoir récupéré ces informations, Le contexte courant de l'utilisateur/Dispositif (Cc) est mis en correspondance sémantiquement avec le contexte du service candidat (CSdi). Le degré de similarité résultant est par la suite stocké dans la variable (Cscore). L'algorithme sélectionne le service comme service candidat. Finalement, si un autre service est toujours dis-

```
Algorithm: Proactive service discovery
1:  Procedure ProactiveCandidateService (Ce, Sd, Lo)   /* Ce: Context Entity, Sd: Service Discover, Lo: Location */
2:  cscore = 0
3:   BEGIN
4:      For each Sd(i) ∈ Sd  do
5:        C(c) = GetContextChangeDetection (Ce)          /* Detect Context Change */
6:        cscore = ContextualMatching (Cc, CSdi)         /* Cc: Context Current */
7:      end For
8:    CPS = GetCandidateService (cscore)                 /* Recuperate the Best Score Of lists Service's */
9:    Return CPS
10: END
11: procedure ContextChangeDetection (Ce)
12: BEGIN
13:      For each Ce (i) ∈ Ce do
14:        Update (Ce)                                   /* Recuperate the Last Change */
15:      End For
16:   Return Ce
17: END procedure
18: Procedure ContextualMatching (Cc, CSdi, Onto)      /* Onto: Ontology of Context Manager*/
19: CtxScore = 0
20:  BEGIN
21:    For each Cc(i) ∈  Cc do
22:      ECci = GetEntity.Value (Cc)                   /* Recuperate Context Entity Value */
23:           For each CSdi(i) ∈  CSdi  do
24:                ECSdi = GetEntity.Value (CSdi)     /* Recuperate Context Service Value */
25:           End for
26:      Escore= ContextMatch(ECci, ECSdi)
27:      CtxScore =: Cscore + Escore
28:    End for
30:    Return CtxScore                                 /* Final Score Of Matching */
31:  END procedure
```

Figure VI.18 – Illustration de l'algorithme de selection des services proactifs.

ponible, alors l'algorithme relance la même démarche décrite ci-dessus. Sinon, l'algorithme sélectionne le service (CPS) ayant le score le plus élevé et le retourne comme résultat final.
Nous détaillons l'algorithme : Contextuel Matchin. Il prend en entrée la description du contexte courant de l'utilisateur (Cc), la description du contexte requis du service (CSdi) et l'ontologie multi-niveaux de contexte. Il produit comme sortie le degré de correspondance ou de satisfaction (Cscore) des conditions contextuelles du service avec contexte courant de l'utilisateur/Dispositif. Ainsi, pour chaque condition contextuelle (Cci), l'algorithme récupère l'entité (ECci) et l'entité de contexte (ECsdi) du service en cours. Par la suite, une mise en correspondance entre les entités (ECci)et (ECsdi) . Un degré de similarité (Escore)est attribué à cette mise en correspondance. Le degré de similarité final des deux entités est calculé selon la formule suivante : Cscore = (Escore + Cscore) et on calcule a chaque fois le nouveu Cscore. Finalement, le score global (Cscore) de satisfaction de toutes les conditions contextuelles de service par rapport aux conditions contextuelles de l'environnement. L'algorithme sort ensuite de la boucle et passe au service suivant. Une fois toutes les conditions vérifiées, l'algorithme se termine et retourne comme résultat final le score de la mise en correspondance contextuelle (Cscore).

6 Cas études

6.1 Plate forme et Expérimentations

Dans cette partie, nous présentons l'implémentation des concepts que nous avons présenté précédemment. Pour cela, nous nous appuyons sur la plate-forme générique définie dans le chapitre 4 que nous avons étendue [KB13b].

6.1.1 Présentation de la plate-forme

Notre plate-forme logicielle est basée sur le Framework .NET qui offre un environnement de qualité : Visual Studio 2010. Nous proposons une extension aux services web comprenant la notion de services web pour Dispositifs dont l'objectif est d'abstraire la communication entre dispositifs sous un standard commun. Nous nous sommes basés sur le standard UPnP. Il remplit les conditions pour correspondre à notre notion de service web pour Dispositif. UPnP est une initiative industrielle conçue pour permettre la connectivité simple et robuste entre dispositifs de différents fournisseurs sur un réseau IP/LAN. L'architecture UPnP permet par sa nature la découverte et le contrôle de dispositifs sur un réseau, indépendamment des systèmes d'exploitation (utilisation des sockets), des langages de programmation (stubs disponibles pour plusieurs langages) et des réseaux (grâce à l'utilisation des normes existantes : IP, UDP, TCP, HTTP, XML et SOAP).

6.1.2 Mise en œuvre d'un espace pervasif de Services pour Dispositifs

Ces dispositifs peuvent être virtuels ou réels dans l'environnement personnel d'un utilisateur ou bien enfouis dans l'environnement physique. L'infrastructure offre ainsi un certain nombre de services variés tels que :

1. Des services UPnP virtuelles d'une scène 2D et 3D dans laquelle l'utilisateur est immergé.
2. Des services UPnP réels utilisant des technologies.
3. Services web standard comme les services météo sur Internet.

Cet environnement permet l'évaluation des nouvelles applications de l'informatique mobile et ambiante telles que les usages des Wearable Computers et la domotique. C'est un cadre d'expérimentation pour le prototypage de ce type d'application. L'architecture de nos exemples s'appuie sur un designer UPnP. C'est un service qui a deux fonctionnalités : créer les composants Proxy des dispositifs UPnP découverts et détruire les composants Proxy des dispositifs UPnP disparus [KB14b].

Afin d'illustrer notre approche, nous présentons un scénario qui mettra en œuvre notre infrastructure dans un environnement pervasif. Le prototype simule un musée virtuel pour assister les visiteurs, Figure 6.19 [KB14b]. L'infrastructure a pour rôle d'assister les visiteurs du musée en leurs proposant des services (vidéo explicatives, historiques) concernant l'œuvre d'art a proximité de lequel il se trouve. Cette application est destinée à être utilisée avec différents appareils mobiles tels que des PDA, des Mobile PC ou des smartphones.

La figure 6.19 présente l'environnement ubiquitaire dans lequel se déroule la partie d'assistance aux visiteurs de ce cas d'études. L'objectif est de proposer des services pertinents basé sur la localisation du visiteur en tenant compte son contexte. Par exemple, lorsqu'un visiteur est dans le musée, il souhaite, pour chacune des œuvre présentées, pouvoir disposer d'informations sur sa création, son auteur, son histoire, Ou bien avoir une vidéo explicatives de l'œuvre d'art à proximité. Il souhaite également pouvoir visiter librement, avec ou sans l'aide d'un plan du musée, ou au contraire être guidé depuis son entrée jusqu'à sa sortie du musée pour ne pas perdre de temps et ne pas rater les œuvres incontournables.

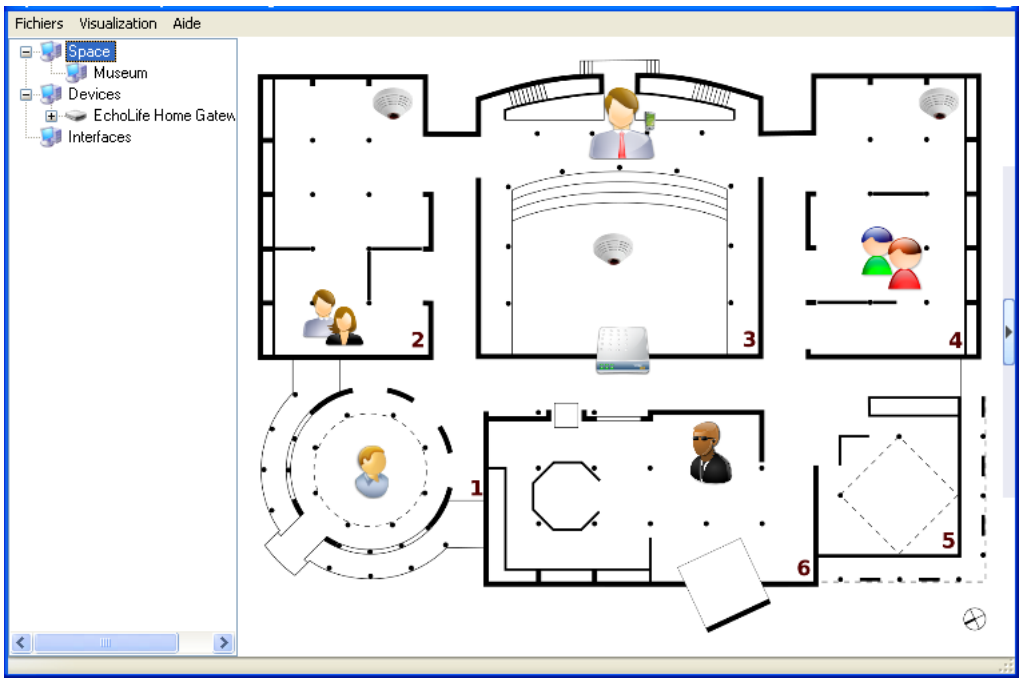

Figure VI.19 – Exploitation du prototype pour assistance aux visiteurs dans un musé.

Service Name	ContextService	ContextEntity	Score
S1	-Location.Museums=Zoological -Profile.Age = 21 -Location.City = SBA -Location.Country = AlGERIA -Resource.Screen = 16 -Device = Intel Core 2 duo -Resource.Memory = 2048 -Resource.Network = Ethernet	 -Location.Museums=Archaeology -Profile.Age = 25 -Profile.Langage = Arabic -Profile.Role = Tourist -Location.City = SBA -Location.Country = AlGERIA -Resource.Network = 3G -Device = Iphone 5 -Resource.Memory = 16 -Resource.Screen = 5	2
S2	-Location.Museums=Archaeology -Profile.Age = 24 -Profile. Langage = Arabic -Profile.Role = Citizen -Location.City = TLEMCEN -Location.Country = ALGERIA, -Resource.Network = 3G -Device = Ipad 2 -Resource.Memory = 32 -Resource.Screen = 5		4
S3	-Location.Museums=Archaeology -Profile.Age = 25 -Profile. Langage = Arabic -Profile.Role = tourist -Location.City = ORAN -Location.Country = ALGERIA -Resource.Network = Wifi -Device = Iphone 5 -Resource.Memory = 16 -Resource.Screen = 5		8

Figure VI.20 – Description de trois services proposés.

La détection des visiteurs est assurée par des capteurs et des lecteurs de Bluetooth qui détecte la présence des périphériques à proximité des œuvres d'arts. Si un visiteur est équipé d'un téléphone mobile ou un assistant personnel,

il peut visualisé les vidéo directement sur son dispositif, sinon notre prototype s'engage à trouver un autre service proactif plus adapté à son dispositif.

Scenario : Amine, est un touriste equipé d'un smartphone (Iphone5), commence à visiter les differents monuments du musé. Le lecteur de Bluetooth du monument concerné detecte la présence du dispositif de Amine (Iphone5). Le gestionnaire de localisation notifie un message de detection qui se traduit par :

- L'ajout de l'individu à la classe Tourist (Sous classe de User), Le dispositif (Iphone5) à la classe Device et le monument (Archaeology) à la classe Location.
- La mise à jour des propietées "isSituatedIn" et "isUsingBy".
- Le gestionnaire de contexte aprés avoir lancer le motuer d'inference va déduire l'œuvre d'art visité par le touriste (Archaeology).

Iphone5(?d), monument(?m), Tourist (?t), isSituatedIn(?d, ?m), isUsingBy (?t, ?d) == IsLocatedIn (?t, ?m).

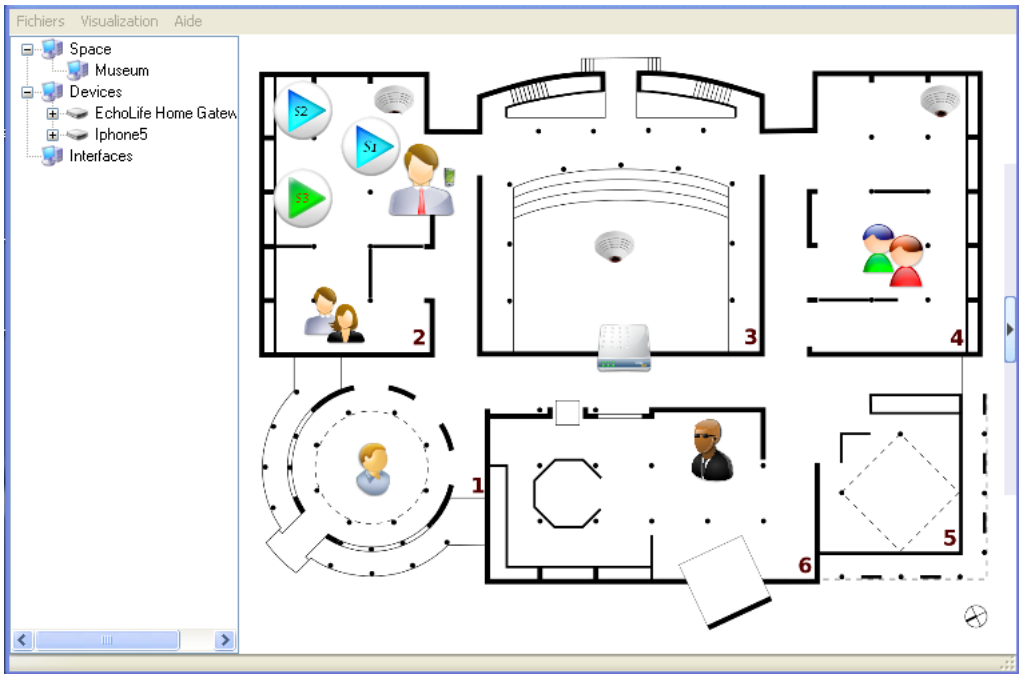

Figure VI.21 – Selection du service pertinent : S3.

Afin de répondre au besoin de Amine qui souhaite profiter des vidéos contextuelles proposées par le musée. L'application de visite de musée selectionne trois services de description, Figure 6.20 : S1, S2, et S3 de la classe "Service" du modéle de contexte et la propritée "IsAvailableOn" est mis à jour. La primitive Lecture (?s, ?desc) est executée par le moteur d'inference, ceci se traduit par une invocation du service de lecture multimedia.

Iphone5(?d), monument(?m), Tourist (?t), description (?desc, ?m), Lecture (?s), IsLocatedIn (?t, ?m), IsAvailableOn (?d, ?s) == Lecture (?s, ?desc).

Pour evaluer la validité de notre algorith : ProactiveCandidateService, l'environement est deployé sur une machine Intel Core i5 1.3 GHz with 4 GB memory. apres l'execution de l'algorithme le prototype selectionne le service approprié au tourist suivant son context courant : S3, Figure 6.21.

7 Conclusion

L'accès aux services demeure trop complexe. En effet, dans un grand nombre de situations, l'utilisateur doit choisir tout seul un service parmi plusieurs implémentations disponibles, sans avoir pour autant le bagage nécessaire pour comprendre ces implémentations. Nous devons lui offrir le service proactive d'une façon implicite qui satisfait ses besoins, sans qu'il soit forcé de comprendre les détails et les contraintes des dispositifs utilisés. Dans ce chapitre, nous avons proposé une approche capable d'apporter à celui-ci des services adaptés à son contexte d'utilisation tout en gardant un niveau de transparence convenable. Afin de démontrer cette approche, nous avons implémenté le processus proposé. Ce processus proactif de découverte de services pertinents dans une infrastructure UPnP.

Chapitre VII

Conclusion générale et perspectives

1 Conclusion

Le travail réalisé dans le cadre de cette thèse se positionne dans le domaine de l'Informatique Pervasive et l'émergence des Systèmes d'Information. Dans ces domaines, nous nous sommes concentrés sur l'émergence d'une nouvelle génération de SI, les Systèmes d'information Pervasifs (SIP). Cette nouvelle génération doit désormais être conçue et mise en œuvre en tant que système sensible au contexte, afin de répondre au mieux aux besoins des utilisateurs.

Afin de répondre à cette problématique, résumée ci-dessus, nous avons proposé, une nouvelle vision d'un SIP transparent, nous nous sommes basés, dans le cadre de cette thèse, sur les notions de contexte, la pro activité et de service.

Cette vision qui se base sur l'orientation service :

1. Permet de répondre au besoin de gestion de l'hétérogénéité technique de l'environnement intégrant ce système, grâce à son indépendance par rapport aux aspects technologiques.
2. Est fondée sur la sensibilité au contexte gérant ainsi la dynamicité de l'environnement pervasif.
3. Est conçue autour d'une orientation proactive laquelle permet de proposer des services d'une manière implicite répondant aux besoins des consommateurs sans autant fournir la moindre requête.

Pour mettre en place cette vision proactive et contextuelle des SIP, nos contributions, se résument autour des points suivants :

- **Une Architecture conceptuelle de SIP :** En étudiant la vision actuelle des systèmes ubiquitaires, nous avons dressé une liste de défis et chalenges. Notre contribution se résume en un modèle architectural générique qui tient en compte des éléments de base d'un environnement ubiquitaire. Nous avons identifié cinq grands domaines qui constituent un environnement pervasif : infrastructure, dispositifs, interfaces, espaces intelligents et sécurité. La couche d'infrastructure intègre l'ensemble technologique utilisé pour la conception des systèmes pervasifs, La couche des objets intègre les dispositifs communicants. La couche interface permet aux usagers de contrôler et interagir avec des objets d'une manière intuitive. La couche espace intelligent repose sur des capacités de perception, d'action et de communication et enfin la couche sécurité.

- **Un Modèle de contexte générique :** La modélisation du contexte est une étape essentielle pour le développement des applications sensibles aux contextes. La plupart des approches proposées dans la littérature sont ou bien spécifiques à une application ou à un domaine particulier ou bien il leur manque le formalisme nécessaire pour la modélisation. Ces approches ne sont pas suffisamment génériques pour être réutilisables dans d'autres applications ou autres domaines et ont un niveau de reconfiguration limité. Ainsi, il est nécessaire de représenter le fonctionnement global du système à travers une modélisation de haut niveau (Ontologie Générique). Les recherches au niveau des systèmes pervasifs se sont essentiellement concentrées sur le niveau technique. Des efforts importants ont été consacrés sur l'adaptation au contexte, surtout à la localisation et aux terminaux.

- **Des mécanismes proactifs de services :** Afin de gérer l'interaction des utilisateurs et des dispositifs dans un environnement ubiquitaire, nous proposons des des services proactifs guidés par la localisation et le contexte. Ces mécanismes permettent de proposer et de satisfaire les besoins de l'utilisateur en lui offrant le service le plus approprié, selon le contexte, assurant ainsi la dynamique et la pro-activité des SIP. Le mécanisme de découverte de services dynamiques est assuré par l'infrastructure UPnP, que nous estimons une solution intéressante pour la conception des futur SIP. Nous avons proposé également un mécanisme proactif de services. Ce mécanisme permet d'anticiper les besoins futurs de l'utilisateur ou du dispositif et de lui suggérer, d'une manière transparente et implicite, le service le plus approprié qui pourra, par la suite, l'intéresser.

2 Perspectives

Ce travail ouvre la voie à de nouvelles perspectives intéressantes. Nous soulignons dans la suite, certaines de ces perspectives qui vont contribuer à l'évolution des propositions que nous avons réalisées dans le cadre de cette thèse. Ces perspectives s'organisent comme suit :

- **Expérimentation de l'architecture en monde réel :** Nous envisageons, à long terme, d'appliquer l'approche proposée en entreprise pour mettre en œuvre le côté pervasif du SI de manière transparente et centrée utilisateur. Ceci nécessite une phase d'expérimentation des différents concepts que nous avons proposé dans le cadre de cette thèse, pour la conception et la construction de SIP.

- **La composition dynamique et l'adaptation des services selon le contexte :** Afin d'étendre notre proposition à un domaine plus ouvert et un système plus dynamique, nous envisageons de faire de la composition de services et de les adapter dynamiquement selon le contexte avant de les proposer explicitement à l'utilisateur ou au diapositifs communicant.
- **mécanisme de sécurité contextuel :** Les opportunités ouvertes par les systèmes pervasifs sont extrêmement nombreuses et prometteuses. Du point de vue de l'utilisateur et de son interaction avec son environnement, la sécurité est un point fondamental, à la fois pour la confidentialité et l'intégrité de ses données ou pour la sureté de ses équipements. Une large adoption des systèmes pervasifs ne peut se faire sans une approche intégrée de la sécurité. Nous envisageons de proposer une architecture pervasive augmentée de différents forme de règles d'accès à l'information, aux services et aux équipements.

Annexe : Liste des Acronymes

ADL : Architecture Description Language.
AC : Ambient Computing.
CA : Context Awareness.
CORBA : Common Object Request Broker Architecture.
CLIPS : C Language Integrated Production System.
DIS : Desktop Information Systems.
DLNA : Digital Living Network Alliance.
DPWS : Device Profile for Web Services.
DSL : Domain-Specific Language.
DPWS : Device Profile for Web Services
EJB : Enterprise JavaBean.
GA4PCsim : Generic Architecture for pervasive Computing Simulator.
GPS : Global Positioning System.
GPRS : General Packet Radio Service
GSM : Global System for Mobile Communications
GUI : Graphical User Interface.
HTTP : HyperText Transfer Protocol.
IDE : Integrated Development Environment.
IDL : Interface Description Language.
IM : Instant Messaging.
IT : Information Technology.
ICT :Information and Communication Technology.
IoS : Internet Of Services.
IoT : Internet Of Things.
OS : Operating System.
OWL : Ontology Web Language.
OOP : Object-Oriented Programming.
PC : Personal Computer.
PDA : Personal Digital Assistant.
PC : Pervasive Computing.

QoS : Quality of Service.
RLAN : Radio Local Area Networks.
RDF : Resource Description Framework.
RFID : Radio Frequency IDentification.
SI : Systémes d'Information.
SIP : Systémes d'Information perfasifs.
SOA : Service Oriented Architecture.
SOAP : Simple Object Access Protocol.
SOC : Service Oriented Computing.
SLP : Service Location Protocol.
SWRL : Sementic Web Rule Language.
UC : Ubiquitous Computing.
UDDI : Universal Description Discovery and Integration.
UI : User Interface.
UML : Unified Modeling Language.
URL : Uniform Resource Locator.
UPnP : Universal Plug Play.
URI : Uniform Resource Identifier.
UMTS : Universal Mobile Telecommunications System.
VANET : Réseau Ad-Hoc de véhicules.
WiMAX : Worldwide Interoperability for Microwave Access.
WLAN : Wireless Local Area Network.
W3C : World Wide Web Consortium.
WSDL : WebService Description Language.
WSDL : WebService Description Language.
WSOAD : Web Service Oriented Architecture For Devices.

Bibliographie

[AAF+96] Gregory D. Abowd, Christopher G. Atkeson, Ami Feinstein, Cindy E. Hmelo, Rob Kooper, Sue Long, Nitin "Nick" Sawhney, and Mikiya Tani. Teaching and learning as multimedia authoring : The classroom 2000 project. In Philippe Aigrain, Wendy Hall, Thomas D. C. Little, and V. Michael Bove Jr., editors, *ACM Multimedia*, pages 187–198. ACM Press, 1996. 54

[AAH+97] Gregory D. Abowd, Christopher G. Atkeson, Jason I. Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide : A mobile context-aware tour guide. *Wireless Networks*, 3(5) :421–433, 1997. 53

[A.B07] R.Ajhoun A.Begdouri, M.Berraho. L'informatique mobile au service des applications medicales. In *ISWC*, pages 1–11, 2007. 20

[ADB+99] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304–307, London, UK, UK, 1999. Springer-Verlag. 30, 38

[AFG+04] Liliana Ardissono, Roberto Furnari, Anna Goy, Giovanna Petrone, and Marino Segnan. A framework for the management of context-aware workflow systems, 2004. 41

[AJG12] Fatma Achour, Anis Jedidi, and Faiez Gargouri. The generic model for pervasive information system. In *Proceedings of the 3rd International Conference on Information and Communication Systems*, ICICS '12, pages 18 :1–18 :7, New York, NY, USA, 2012. ACM. 19, 80, 81

[ARC10] Imad Afyouni, Cyril Ray, and Christophe Claramunt. Indoor space modelling, mobility, and positioning in ubiquitous information systems. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, pages 33–38. ACM, 2010. 80, 81

[AV12] V. Aggarwal and J. Vaidya. Customizing mobile applications, October 4 2012. US Patent App. 13/077,599. 45

[BB06] Nabiha Belhanafi-Behlouli. *Ajout de mecanismes de reactivite au contexte dans les intergiciels pour composants dans le cadre d'utilisateurs nomades*. PhD thesis, 2006. These de doctorat dirigee par

Bernard, Guy Informatique Evry, Institut national des telecommunications 2006. 35, 39, 42, 44, 45, 46, 50, 51

[BBC97] P.J. Brown, J.D. Bovey, and Xian Chen. Context-aware applications : from the laboratory to the marketplace. *Personal Communications, IEEE [see also IEEE Wireless Communications]*, 4(5) :58–64, 1997. 29, 37, 38, 41

[BBC+03] David F. Bantz, Chatschik Bisdikian, David Challener, John P. Karidis, Steve Mastrianni, Ajay Mohindra, Dennis G. Shea, and Michael Vanover. Autonomic personal computing. *IBM Systems Journal*, 42(1) :165–176, 2003. 29

[BC12] Ansem Ben Cheikh. *E-CARe : a method for ubiquitous Information Systems Engineering*. Theses, Université de Grenoble, June 2012. 16

[BCMT06] Paolo Bellavista, A. Corradi, Rebecca Montanari, and Alessandra Toninelli. Context-aware semantic discovery for next generation mobile systems. *IEEE Communications Magazine*, 48(9) :62–71, 2006. 112

[BcP99] P. Brezillon and J. ch. Pomerol. Contextual knowledge sharing and cooperation in intelligent assistant systems. *Le Travail Humain*, 62 :223–246, 1999. 38

[BD07] Genevieve Bell and Paul Dourish. Yesterday's tomorrows : notes on ubiquitous computing's dominant vision. *Personal and Ubiquitous Computing*, 11(2) :133–143, 2007. 4

[BGL07] Andre Bottaro, Anne Gerodolle, and Philippe Lalanda. Pervasive service composition in the home network. In *AINA*, pages 596–603. IEEE Computer Society, 2007. 106

[BHM+04] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Mike Champion, Christopher Ferris, and David Orchard. Web services architecture. World Wide Web Consortium, Note NOTE-ws-arch-20040211, February 2004. 104

[BMPG+07] Sonia Ben Mokhtar, Davy Preuveneers, Nikolaos Georgantas, Valérie Issarny, and Yolande Berbers. EASY : Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support. *Journal of Systems and Software*, 81(5) :785–808, 2007. 112

[Bou08] Johann Bourcier. *Auto-Home : une plate-forme pour la gestion autonomique d'applications pervasives*. PhD thesis, UniversitÃl' Joseph Fourier, Grenoble, November 2008. 26, 27, 28, 30, 84

[BOU10] Samia BOUARROUDJ. *Resonement sur une ontologie enrichie par des regles SWRL pour la recherche semantuque d'mages annotees*. PhD thesis, UNIVERSITE 20 AOUT 1955 SKIKDA, November 2010. 91, 92, 93

[Box97] Don Box. *Essential COM*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1997. 64

[Bro96] P.J. Brown. The stick-e document : A framework for creating context-aware applications. In *Proceedings of the Electronic Publishing*, pages 259–272, Laxenburg, Austria, September 1996. 29, 32, 37, 40, 46

[CBCP01] Michael Clarke, Gordon S. Blair, Geoff Coulson, and Nikos Parlavantzas. An efficient component model for the construction of adaptive middleware. In Rachid Guerraoui, editor, *Middleware*, volume 2218 of *Lecture Notes in Computer Science*, pages 160–178. Springer, 2001. 64

[CC08] Joëlle Coutaz and James Crowley. Plan "intelligence ambiante" : Défis et opportunités. Document de réflexion conjoint du comité d'experts " Informatique Ambiante " du département ST2I du CNRS et du Groupe de Travail " Intelligence Ambiante " du Groupe de Concertation Sectoriel (GCS3) du Ministère de l'Enseignement Supérieur et de la Recherche, DGRI A3., 2008. 42

[CDB+12] Marco Conti, Sajal K. Das, Chatschik Bisdikian, Mohan Kumar, Lionel M. Ni, Andrea Passarella, George Roussos, Gerhard TrÃűster, Gene Tsudik, and Franco Zambonelli. Looking ahead in pervasive computing : Challenges and opportunities in the era of cyber-physical convergence. *Pervasive and Mobile Computing*, 8(1) :2–21, 2012. 80, 81

[CDS04] Dan Chalmers, Naranker Dulay, and Morris Sloman. Towards reasoning about context in the presence of uncertainty. In *Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp 2004 Nottingham, UK*, 2004. 38

[CFJ03] Harry Chen, Tim Finin, and Anupam Joshi. An ontology for context-aware pervasive computing environments. In *Workshop on Ontologies and Distributed Systems*. IJCAI-2003, August 2003. 58, 59, 61, 93

[CFW09] Daniel Cheung-Foo-Wo. *Adaptation Dynamique par Tissage d'Aspects d'Assemblage*. PhD thesis, Université de Nice - Sophia Antipolis, March 2009. 28

[Cha07] Tarak Chaari. *Adaptation d'applications pervasives dans des environnements multi-contextes*. These de doctorat en informatique, INSA de Lyon, September 2007. 39, 41, 42, 44, 50, 51, 55, 61, 67

[CJFY05] Dipanjan Chakraborty, Anupam Joshi, Timothy W. Finin, and Yelena Yesha. Service composition for mobile environments. *MONET*, 10(4) :435–451, 2005. 106

[CK00] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000. 37, 38

[CK05] Eleni Christopoulou and Achilles Kameas. Gas ontology : An ontology for collaboration among ubiquitous computing devices. *Int. J. Hum.-Comput. Stud.*, 62(5) :664–685, 2005. 93

[Cos08] C Costa. *Continuum : A context-aware service-based software infrastructure for ubiquitous computing*. PhD thesis, UFRGS, 2008. 27

[CPFJ04] Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi. SOUPA : Standard Ontology for Ubiquitous and Pervasive Applications. In *International Conference on Mobile and Ubiquitous Systems : Networking and Services*, Boston, MA, August 2004. 93

[CSKL13] Chii Chang, Satish Narayana Srirama, Shonali Krishnaswamy, and Sea Ling. Proactive web service discovery for mobile social network in proximity. 4, 2013. 112

[CV09] Karlene Cousins and Upkar Varshney. Designing ubiquitous computing environments to support work life balance. *Commun. ACM*, 52(5) :117–123, 2009. 30, 80, 81

[DAS01] Anind K. Dey, Gregory D. Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human - Computer Interaction*, 16(2, 3 & 4) :97–166, 2001. 38, 40, 41, 42, 55, 61, 97, 115

[Dej07] Ejigu Dejene. *Services Pervasifs Contextualises : Modelisation et Services Pervasifs Contextualises : Modelisation et*. These de doctorat en informatique, INSA Lyon, December 2007. 36

[Dey00] Anind K. Dey. *Providing architectural support for building context-aware applications*. PhD thesis, Georgia Institute of Technology, 2000. 38, 41

[Dey01] Anind K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1) :4–7, 2001. 30

[DFAB97] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human-computer Interaction*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997. 24

[DIBSS04] Hector A. Duran-limon, Gordon S. Blair, Thirunavukkarasu Sivaharan, and George Samartzidis. Resource management for the real-time support of an embedded publish/subscribe system, 2004. 64

[DM09] Dan Driscoll and Antoine Mensch. Devices Profile for Web Services Version 1.1. Technical report, July 2009. 110, 111

[DRW06] Leticia Duboc, David S. Rosenblum, and Tony Wicks. A framework for modelling and analysis of software systems scalability. In Leon J. Osterweil, H. Dieter Rombach, and Mary Lou Soffa, editors, *ICSE*, pages 949–952. ACM, 2006. 27, 28

[DSAM] Hand Held Devices, Shahid Siddiq, Aasim Ali, and Kamran Malik. Gap analysis between ubiquitous computing requirements and features of an open source operating system (openmoko) for. 26

[Dug07] Pascal Dugenie. *Espaces Collaboratifs Ubiquitaires sur une infrastructure à ressources distribuées*. Theses, Université Montpellier II - Sciences et Techniques du Languedoc, December 2007. 26

[en.12] en.wikipedia.org. Autonomy corporation - wikipedia, the free encyclopedia, 2012. 28

[Erl05] Thomas Erl. *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005. 104

[Esc08] Clement Escoffier. *iPOJO : A flexible service-oriented component model for dynamic systems*. Theses, Université Joseph-Fourier - Grenoble I, December 2008. 16, 27, 31

[ESY04] Natalia Em, Euiho Suh, and Keedong Yoo. Extracting requirements for ubiquitous computing technology-based is using factor analysis. In Jian Chen, editor, *ICEB*, pages 768–770. Academic Publishers/World Publishing Corporation, 2004. 80, 81

[FC04] Patrick Fahy and Siobhan Clarke. Cass a middleware for mobile context-aware applications. In *Workshop on Context Awareness, MobiSys*, 2004. 62

[FFL11] Giuseppe Fenza, Domenico Furno, and Vincenzo Loia. Enhanced healthcare environment by means of proactive context aware service discovery. In *AINA*, pages 625–632. IEEE Computer Society, 2011. 111

[FGM05] Areski Flissi, Christophe Gransart, and Philippe Merle. Une infrastructure Ãă composants pour des applications ubiquitaires. In JoÃńlle Coutaz and Sylvain Lecomte, editors, *UbiMob*, ACM International Conference Proceeding Series, pages 45–48. ACM, 2005. 28

[FLRT08] Nicolas Ferry, Stéphane Lavirotte, Gaëtan Rey, and Jean-Yves Tigli. Adaptation Dynamique d'Applications au Contexte en Informatique Ambiante. Technical Report I3S/RR-2008-20-FR, I3S (Université de Nice - Sophia Antipolis / CNRS), Sophia Antipolis, France, October 2008. 28, 107

[Fon12] Emeric Fontaine. *Programmation d'espace intelligent par l'utilisateur final*. Theses, Université de Grenoble, July 2012. 84

[FTL+11] Nicolas Ferry, Jean-Yves Tigli, Stephane Lavirotte, Gaëtan Rey, and Michel Riveill. Wcomp, a middleware for ubiquitous computing. *CoRR*, abs/1111.1904, 2011. 103, 104, 108

[FWK02] P. Fremantle, S. Weerawarana, and R. Khalaf. Enterprise services. *Communications of the ACM*, 45(10) :77–82, 2002. 74

[Gal12] Mathieu Gallissot. *Modelling the concept of comfort within a smart building : from senses to behaviour*. Theses, Université de Grenoble, April 2012. 24

[GDG10] Carlos Granell, Laura Diaz, and Michael Gould. Service-oriented applications for environmental models : Reusable geospatial services. *Environmental Modelling and Software*, 25(2) :182–198, 2010. 75

[GKPMD13] Kimberly Garcia, Manuele Kirsch-Pinheiro, Sonia Mendoza, and Dominique Decouchant. An ontological model for resource sharing in pervasive environments. In *Web Intelligence*, pages 179–184. IEEE Computer Society, 2013. 93

[GM04] Puneet Gupta and Deependra Moitra. Evolving a pervasive it infrastructure : A technology integration approach. *Personal Ubiquitous Comput.*, 8(1) :31–41, February 2004. 80, 81

[GPZ05] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A service-oriented middleware for building context-aware services. *J. Netw. Comput. Appl.*, 28(1) :1–18, January 2005. 65, 66, 93

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2) :199–220, June 1993. 91

[GWPZ04] Tao Gu, Xiao Hang Wang, Hung Keng Pung, and Da Qing 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. 93

[HBS02] A. Held, S. Buchholz, and A. Schill. Modeling of context information for pervasive computing applications. In *Proc. of the 6th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, FL, July 2002. 47

[Hef04] Jeff Heflin. OWL Web Ontology Language - Use Cases and Requirements. W3c recommendation, World Wide Web Consortium (W3C), 10 February 2004. http ://www.w3.org/TR/2004/REC-webont-req-20040210/. 48

[HI04] K. Henricksen and J. Indulska. Modelling and using imperfect context information, 2004. 38

[Hoa07] Didier Hoareau. *Ubiquitous components for dynamic networks*. Theses, Université de Bretagne Sud, December 2007. 27, 28

[HPSB+04] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. Swrl : A semantic web rule language combining owl and ruleml. W3c member submission, World Wide Web Consortium, 2004. 98, 99

[HSP+03] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann, and Werner Retschitzegger. Context-awareness on mobile devices - the hydrogen approach. In *HICSS*, page 292, 2003. 46

[HTH06] S. Hoh, J.S. Tan, and M. Hartley. Context-aware systems a primer for user-centred services. *BT Technology Journal*, 24(2) :186–194, 2006. 80, 81

[idn96] Organisation internationale de normalisation. *Technologies de l'information : traitement réparti ouvert : modèle de référence : Fondements. –*. Number ptie. 1 in Normes ISO/CEI. 1996. 32

[IRRH03] Jadwiga Indulska, Ricky Robinson, Andry Rakotonirainy, and Karen Henricksen. Experiences in using CC/PP in context-aware systems. In *4th International Conference on Mobile Data Management (MDM)*, volume 2574 of *Lecture Notes in Computer Science*, pages 247–261. Springer, January 2003. 46

[Jam01] Anthony Jameson. Modelling both the context and the user. *Personal Ubiquitous Comput.*, 5(1) :29–33, January 2001. 29

[Jou09] Wilfried Jouve. *Approche déclarative pour la génération de canevas logiciels dédiés à l'informatique ubiquitaire*. Theses, Université Sciences et Technologies - Bordeaux I, April 2009. 28

[Kac09] Rahim Kacimi. Techniques de conservation d'énergie pour les réseaux de capteurs sans fil. September 2009. 21, 22

[KB13a] Mohammed Fethi Khalfi and Sidi Mohamed Benslimane. Toward a Generic infrastructure for Ubiquitous Computing. In *3rd International Workshop on Pervasive and Context-Aware Middleware (PerCAM 2013) in conjunction with Fourth International Joint Conference on Ambient Intelligence (AmI 2013)*, pages –, Dublin, Ireland, December 2013. 18

[KB13b] Mohammed Fethi Khalfi and Sidi Mohamed Benslimane. Toward a generic infrastructure for ubiquitous computing. *Int. J. Adv. Pervasive Ubiquitous Comput.*, 5(1) :66–85, January 2013. 81, 82, 83, 84, 86, 103, 122

[KB14a] Mohammed Fethi Khalfi and Sidi Mohamed Benslimane. A framework for ambient computing. In *CLOSER 2014 - Proceedings of the 4th International Conference on Cloud Computing and Services Science, Barcelona, Spain, April 3-5, 2014.*, pages 170–178, 2014. 79, 80, 81, 86

[KB14b] Mohammed Fethi Khalfi and Sidi Mohamed Benslimane. Proactive Approach for Service Discovery Using Web Service for Devices On Pervasive Computing. In *The 4th International Workshop on Pervasive and Context-Aware Middleware (PerCAM 14), Dubai, UAE on October 16, 2014.*, pages –, Dubai, UAE, October 2014. 79, 86, 95, 96, 100, 106, 112, 113, 114, 115, 116, 119, 122

[KB14c] Mohammed Fethi Khalfi and Sidi Mohamed Benslimane. Systemes d'information pervasifs : Architecture et challenges. In *UbiMob2014 : 10'emes journ'ees francophones Mobilit'e et Ubiquit'e 5-6 juin 2014 Sophia Antipolis (France)*, pages –, 2014. 19, 26

[KC04] Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF) : Concepts and abstract syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, February 2004. 48

[Ker11] Serge Kernbach. Challenges of Pervasive Adaptation : Viewpoint from Robotics. In Alois Ferscha, editor, *Pervasive Adaptation - The Next Generation Pervasive Computing Research Agenda*, pages 43+. Institute for Pervasive Computing, Johannes Kepler University, 2011. 3

[KFNM04] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. The protege owl plugin : An open development environment for semantic web applications. *in press*, 2004. 96

[KG06] Panos E. Kourouthanassis and George M. Giaglis. A design theory for pervasive information systems. In Soraya Kouadri Mostefaoui, Zakaria Maamar, and George M. Giaglis, editors, *IWUC*, pages 62–70. INSTICC Press, 2006. 4, 5, 6, 33

[KGD+06] Mark Keith, Michael Goul, Haluk Demirkan, Jason Nichols, and Margaret C. Mitchell. Contextualizing knowledge management readiness to support change management strategies. In *HICSS*. IEEE Computer Society, 2006. 74

[KMK+03] Panu Korpipaa, Jani Mantyjarvi, Juha Kela, Heikki Keranen, and Esko-Juhani Malm. Managing context information in mobile devices. *IEEE Pervasive Computing*, 2(3) :42–51, 2003. 61, 62

[Kou05] Nabil Kouici. *Gestion des deconnexions pour applications reparties a base de composants en environnements mobiles*. Theses, Institut National des Télécommunications, November 2005. 27

[Kwo12] Joonhee Kwon. A study on effective proactive service in pervasive computing environment. *IJCSNS International Journal of Computer Science and Network Security*, 12(8), 2012. 112

[Laf07] Frederique Laforest. *De l'adaptation a la prise en compte du contexte : une contribution aux systemes d'information pervasifs*. Habilitation a diriger des recherches, INSA Lyon et UniversitÃľ Lyon I, December 2007. 29

[Li10] Boxiao Li. Design and Development of an "Intelligent Environment". Master's thesis, Lehrstuhl für Medientechnik, Technische Universität München, 2010. 26

[Liu06] J. Liu. *Découverte de Services Sensible À la Qualité de Service Dans Les Environnements de L'informatique Diffuse*. 2006. 29

[LL08] Se-Il Lee and Sang-Yong Lee. Integration of user profiles and real-time context information reflecting time-based changes for the recommendation system. pages 270–275, 2008. 45

[Lou10] Christine Louberry. *KAMILUCHO : Context Adaptation for Quality of Service Management*. Theses, Université de Pau et des Pays de l'Adour, September 2010. 27, 28, 32, 40

[Lou12] Mehdi Loukil. *Context management for network access optimisation and services adaptation in heterogeneous environments*. Theses, Institut National des Télécommunications, December 2012. 25

[LS99] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3c recommendation, W3C, February 1999. 92

[McC93] John McCarthy. Notes on formalizing contexts. In Ruzena Bajcsy, editor, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 555–560, San Mateo, California, 1993. Morgan Kaufmann. 48

[Mir09] Moeiz Miraoui. *Architecture Logicielle Pour L'Informatique Diffuse : Modelisation Du Contexte Et Adaptation Dynamique Des Services*. PhD thesis, 2009. AAINR50491. 80, 84

[MPRB04] Ghita Kouadri MostÃľfaoui, Jacques Pasquier-Rocha, and Patrick BrÃľzillon. Context-aware computing : A guide for the pervasive computing community. In *ICPS*, pages 39–48. IEEE Computer Society, 2004. 43

[MS03] Friedemann Mattern and Peter Sturm. From distributed systems to ubiquitous computing - the state of the art, trends, and prospects of future networked systems. In *In K. Irmscher and K.-P. FÃďhnrich, editors, Proc. KIVS 2003*, pages 3–25. Springer-Verlag, 2003. 23

[NBR97] Richard Hull 0002, Philip Neaves, and James Bedford-Roberts. Towards situated computing. In *ISWC*, pages 146–153. IEEE Computer Society, 1997. 41

[NKPSS12] Salma Najar, Manuele Kirsch Pinheiro, Luiz Angelo Steffenel, and Carine Souveyet. Analyse des mécanismes de découverte de services avec prise en charge du contexte et de l'intention. In *8èmes Journées Francophones Mobilité et Ubiquité (Ubimob 2012)*, pages 210–221, Anglet, France, June 2012. Cépaduès Editions. 27, 28

[NKS12] Evangelos Niforatos, Evangelos Karapanos, and Spyros Sioutas. Plbsd : a platform for proactive location-based service discovery. *Journal of Location Based Services*, 6(4) :234–249, 2012. 112

[Org06] Organization for the Advancement of Structured Information Standards. *Reference Model for Service Oriented Architecture 1.0*. OASIS, July 2006. 75

[Pas98] J. Pascoe. Adding generic contextual capabilities to wearable computers. In *ISWC*, pages 92–99, 1998. 29, 38, 46, 52

[PdBW+04] Davy Preuveneers, Jan Van den Bergh, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, Yolande Berbers, Karin Coninx, Viviane Jonckers, and Koen De Bosschere. Towards an extensible context ontology for ambient intelligence. In Panos Markopoulos, Berry Eggen, Emile Aarts, and James L. Crowley, editors, *Second European Symposium on Ambient Intelligence*, volume 3295 of *LNCS*, pages 148 – 159, Eindhoven, The Netherlands, Nov 8 – 11 2004. Springer. 93

[Pet10] Mathieu Petit. *A spatial approach to execution-context modeling in ubiquitous informations systems*. Theses, Ecole nationale supérieure d'arts et métiers - ENSAM, June 2010. 26

[PG03] M. P. Papazoglou and D. Georgakopoulos. Introduction : Service-oriented computing. *Commun. ACM*, 46(10) :24–28, October 2003. 74

[PH07] Mike P. Papazoglou and Willem-Jan Heuvel. Service oriented architectures : Approaches, technologies and research issues. *The VLDB Journal*, 16(3) :389–415, July 2007. 74

[PLZ04] Amir Padovitz, Seng Wai Loke, and Arkady B. Zaslavsky. Towards a theory of context spaces. In *PerCom Workshops*, pages 38–42. IEEE Computer Society, 2004. 111

[PR04a] Nicolas Plouznikoff and Jean-Marc Robert. Caract&#233 ;ristiques, enjeux et d&#233 ;fis de l'informatique port&#233 ;e. In *Proceedings of the 16th Conference on Association Francophone D'Interaction Homme-Machine*, IHM 2004, pages 125–132, New York, NY, USA, 2004. ACM. 27

[PR04b] Nicolas Plouznikoff and Jean-Marc Robert. Caracteristiques, enjeux et defis de l'informatique portee. In Monique Noirhomme-Fraiture, editor, *IHM*, ACM International Conference Proceeding Series, pages 125–132. ACM, 2004. 27, 83

[PTDL07] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing : State of the art and research challenges. *Computer*, 40(11) :38–45, November 2007. 74, 103

[PvBMH09] Pravin Pawar, Bert-Jan van Beijnum, Hailiang Mei, and Hermie Hermens. Towards proactive context-aware service selection in the geographically distributed remote patient monitoring system. In *Proceedings of the 4th International Symposium on Wireless Pervasive Computing, ISWPC 2009*, pages 1–8. IEEE Computer Society Press, February 2009. 111

[RC02] Manuel Roman and Roy H. Campbell. A user-centric, resource-aware, context-sensitive, multi-device application framework for ubiquitous computing environments. page 22, 2002. 57

[RCK10] Mohsen Rouached, Shafique Chaudhry, and Anis Koubaa. Lowpans meet service-oriented architecture. *JUSPN*, 1(1) :39–48, 2010. 103

[Rei04] R. Reix. *Systèmes d'information et management des organisations*. Gestion : Vuibert. Vuibert, 2004. 4

[Rei10] Patrick Reignier. *Intelligence Ambiante Pro-Active : de la Specification a l'Implementation*. Habilitation à diriger des recherches, Université Joseph-Fourier - Grenoble I, September 2010. 27

[RHC+02] Manuel Román, Christopher K. Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt. Gaia : A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing*, pages 74–83, Oct–Dec 2002. 57, 93, 112

[Rod11] Natalia Diaz Rodriguez. A framework for context-aware applications for smart spaces. In *Proceedings of the 2011 IEEE/IPSJ International Symposium on Applications and the Internet*, SAINT '11, pages 218–221, Washington, DC, USA, 2011. IEEE Computer Society. 68

[RPM97] N. Ryan, J. Pascoe, and D. Morse. Enhanced reality fieldwork : the context-aware archaeological assistant. In V. Gaffney, M. van Leusen, and S. Exxon, editors, *Computer Applications and Quantitative Methods in Archaeology (CAA 97), Oxford*, 1997. 38

[Saa09] R. Saadi. *The Chameleon : un système de sécurité pour utilisateurs nomades en environnements pervasifs et collaboratifs*. 2009. 27, 31

[San10] German Sancho. *Adaptation d'architectures logicielles collaboratives dans les environnements ubiquitaires. Contribution à l'interopérabilité par la sémantique*. Theses, Université des Sciences Sociales - Toulouse I, December 2010. 28, 84

[Sar10] I. Sarr. *Routage des transactions dans les bases de données à large échelle*. 2010. 27, 28

[Sar12] Fabien Sartor. *Modeling of interoperability of communicating object their cooperation : implementation to home automation*. Theses, Université de Grenoble, July 2012. 82

[SAT+99] Albrecht Schmidt, Kofi Asante Aidoo, Antti Takaluoma, Urpo Tuomela, Kristof Van Laerhoven, and Walter Van de Velde. Advanced interaction in context. In Hans-Werner Gellersen, editor, *HUC*, volume 1707 of *Lecture Notes in Computer Science*, pages 89–101. Springer, 1999. 38

[Sat01] Mahadev Satyanarayanan. Pervasive computing : vision and challenges. *IEEE Personal Commun.*, 8(4) :10–17, 2001. 16, 42, 80, 81, 84

[SAW94] Bill N. Schilit, Norman I. Adams, and Roy Want. Context-aware computing applications. pages 85–90, December 1994. 29, 37, 38, 40, 41

[SB05] Quan Z. Sheng and Boualem Benatallah. Contextuml : a uml-based modeling language for model-driven development of context-aware web services. In *In : The 4th International Conference on Mobile Business*, pages 206–212, 2005. 45

[SBBC04] Aline Senart, MÃľlanie Bouroche, Gregory Biegel, and Vinny Cahill. A component-based middleware architecture for sentient computing. In *In proc. Of the 2004 ecoop workshop on component-oriented approaches to context-aware computing*, 2004. 64

[Sch10] Albrecht Schmidt. Ubiquitous computing : Are we there yet ? *IEEE Computer*, 43(2) :95–97, February 2010. 4, 45

[Sen07] P. Senn. Objets communicants et nanotechnologie. In *Journees Scientifiques du CNRFS Nanosciences et Radioelectricite*, 2007. 23

[SLP04] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *In : Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004. 93

[SM03] Debashis Saha and Amitava Mukherjee. Pervasive computing : A paradigm for the 21st century. *IEEE Computer*, 36(3) :25–31, 2003. 5, 28, 80, 81

[Sou10] B. Soukkarieh. *Technique de l'internet et ses langages : vers un système d'information web restituant des services web sensibles au contexte*. 2010. 42

[SS10] Aasim Alin Shahid Siddiq. Gap analysis between ubiquitous computing requirements and features of an open source operating system (openmoko) for hand held devices. *International Journal of Electrical Computer Sciences IJECS-IJENS*, 10(5) :15–21, October 2010. 80, 81

[SXX+03] Yuanchun Shi, Weikai Xie, Guangyou Xu, Runting Shi, Enyi Chen, Yanhua Mao, and Fang Liu. The smart classroom : Merging technologies for seamless tele-education. *IEEE Pervasive Computing*, 2(2) :47–55, 2003. 54

[Ten00] David Tennenhouse. Proactive computing. *Commun. ACM*, 43(5) :43–50, May 2000. 111

[TLR+09] Jean-Yves Tigli, Stephane Lavirotte, GaÃńtan Rey, Vincent Hourdin, Daniel Cheung-Foo-Wo, Eric Callegari, and Michel Riveill. Wcomp middleware for ubiquitous computing : Aspects and composite event-based web services. *Annales des TÃľlÃľcommunications*, 64(3-4) :197–214, 2009. 27

[TvS08] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems : Principles and Paradigms*. Prentice Hall International, 2nd rev. ed. edition, 2008. 26

[UG96] M. Uschold and M. Gruninger. Ontologies : Principles, methods and applications. *Knowledge Engineering Review*, 11(2) :93–155, June 1996. 48

[UGUG96] Mike Uschold, Michael Gruninger, Mike Uschold, and Michael Gruninger. Ontologies : Principles, methods and applications. *Knowledge Engineering Review*, 11 :93–136, 1996. 92

[UPn00] UPnP Forum. UPnP Device Architecture. http://www.upnp.org/download/UPnPDA10_20000613.htm, 2000. 108, 109, 111

[VHT00] Antonio Vallecillo, Juan Hernández, and José M. Troya. Component interoperability. Technical Report ITI-2000-37, Universidad de Málaga, Spain, July 2000. 27

[VLA01] Kristof Van Laerhoven and Kofi Aidoo. Teaching context to applications. *Personal Ubiquitous Comput.*, 5(1) :46–49, January 2001. 30

[VVV08] Antti-Matti Vainio, Miika Valtonen, and Jukka Vanhala. Proactive fuzzy control and adaptation methods for smart homes. *IEEE Intelligent Systems*, 23(2) :42–49, 2008. 111

[Wei91] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3) :66–75, January 1991. 4, 5, 7, 15, 16, 17, 19, 24, 29, 35

[WHFG92] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1) :91–102, 1992. 51

[WJH97] Andy Ward, Alan Jones, and Andy Hopper. A new location technique for the active office. *IEEE Personal Commun.*, 4(5) :42–47, 1997. 38

[ws414] ws4d. Device profil for web services., 2014. 107

[WSA+95] Roy Want, Bill N. Schilit, Norman Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis, and Mark Weiser. An overview of the parctab ubiquitous computing experiment. *IEEE Personal Commun.*, 2(6) :28–43, 1995. 52

[WZGP04] X.H. Wang, D.Q. Zhang, T. Gu, and H.K. Pung. Ontology based context modeling and reasoning using owl. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 18– 22, March 2004. 93

[Xia03] Wang Xiaohang. The context gateway : A pervasive computing infrastructure for context aware services, 2003. 41

[XSXX01] Weikai Xie, Yuanchun Shi, Guangyou Xu, and Dong Xie. Smart classroom - an intelligent environment for tele-education. In Heung-Yeung Shum, Mark Liao, and Shih-Fu Chang, editors, *IEEE Pacific Rim Conference on Multimedia*, volume 2195 of *Lecture Notes in Computer Science*, pages 662–668. Springer, 2001. 54

[YK01] Stephen S. Yau and Fariaz Karim. Context-sensitive middleware for real-time software in ubiquitous computing environments. In *ISORC*, pages 163–170. IEEE Computer Society, 2001. 47

[Zai09] Sofia Zaidenberg. *Reinforcement Learning of Context Models for Ubiquitous Computing*. Theses, Institut National Polytechnique de Grenoble - INPG, October 2009. 24

[ZGYR10] Jiehan Zhou, Ekaterina Gilman, Mika Ylianttila, and Jukka Riekki. Pervasive service computing : Visions and challenges. In *CIT*, pages 1335–1339. IEEE Computer Society, 2010. 80, 81

[ZMN05] Fen Zhu, M. W. Mutka, and L. M. Ni. Service discovery in pervasive computing environments. *Pervasive Computing, IEEE*, 4(4) :81–90, 2005. 32