République Algérienne Démocratique et Populaire

Ministère de L'enseignement Supérieur et de la Recherche Scientifique

Université Mouloud Mammeri Tizi-Ouzou

Faculté de Génie Electrique et Informatique

Département d'Informatique

Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master Académique

en informatique

Thème :

L'informatique pervasive (ubiquitaire) : enjeux et perspectives.

Promotion 2013/2014

Option : systèmes informatiques.

Réalise par : FERRAH Mazigh **dirigé par : Mr. Idir RASSOUL, MCA.**

Membres de jury

- **Mr. OUAMRANE Mouhamed** **Maitre assistant classe A, président.**
- **Mr. RASSOUL Idir** **Maitre de conférences classe A, Rapporteur.**
- **Mr. DJEMA Lounas** **Maitre assistant classe A, Examinateur.**
- **Mr. DJAMAH Baouz** **Maitre assistant classe A, Examinateur.**

DEDICACES

A mes très chers parents qui ont sacrifié beaucoup pour moi et grâce à leurs amours et leurs veilles que j'ai pu arriver à ce stade.
A mes chers frères Lyes, Hamza Et Boudjemaa .
A ma chère sœur Malika.
A mon frère Boussad et sa femme Nadia et leurs fille Nelia.
A ma sœur Lamia et son mari Mouhand et leurs enfants Mellissa et Tahar.
Mes amis qui m'ont soutenu en particulier Lydia.

Sommaire :

Chapitre IV : Les architectures des applications ubiquitaires

Chapitre V : Les enjeux de l'informatique ubiquitaires.

Chapitre VI : Conclusion et perspectives

Résumé :

L'informatique ubiquitaire a été l'accès à l'information par le grand public en tout lieu. Il devient aujourd'hui une réalité, grâce à la mise en réseau d'un nombre croissant de dispositifs informatiques par le biais de technologies réseaux et de l'Internet et des ordinateurs de plus en plus nombreux, en particulier les ordinateurs portables et les téléphones de troisième génération

Ce travail de recherche s'inscrit dans le domaine naissant de l'informatique pervasive dont le thème porte essentiellement sur les enjeux et perspectives de (ubiquitaire).

Dans ce mémoire, nous avons présenté l'informatique ubiquitaire, son importance, ses caractéristiques, les technologies, les équipements utilisés et les différentes perspectives de l'informatique ubiquitaire

Mots clés : l'informatique pervasive ; L'informatique ubiquitaire, l'architecture des applications ubiquitaires, La sensibilité au contexte, capteur

1. Introduction générale

Depuis l'aube des temps, l'esprit perfectionniste de l'homme n'a cessé de lui permettre d'améliorer sa vie quotidienne. Le passage de la mécanique aux domaines informatique, électronique et automatique a révolutionné la vie quotidienne de l'être humain. Les nouvelles technologies de l'information et de communication illustrent cette évolution.

La première vague de l'informatique ubiquitaire a été l'accès à l'information par le grand public en tout lieu. Cela a été rendu possible par des ordinateurs de plus en plus nombreux, en particulier les ordinateurs portables et les téléphones de troisième génération. Après, plusieurs domaines s'orientent de plus en plus vers l'adoption des applications ubiquitaires pour gagner la fidélité de leurs clients en leur proposant des services mobiles et intelligents qui nécessitent de moindres efforts de la part de l'utilisateur. Ces applications considèrent le contexte de la personne cible pour proposer des services personnalisés concernant différents domaines comme la santé, les loisirs, la gestion des catastrophes naturelles, les banques, le transport et le domaine militaire.

Les évolutions des équipements matériels permettent aujourd'hui l'informatique ubiquitaire, notamment les réseaux sans fil, les réseaux de capteurs, les équipements informatiques de type domotique ou mobile et a faibles consommation électrique, les étiquettes RFID et les robots. Les objets informatiques se cachent dans des objets de la vie courante : les lampadaires et les conteneurs urbains, les étiquettes RFID colles sur des moutons ou des arbres, participant ainsi a construire l'informatique pervasive dans laquelle le monde réel et le monde virtuel se mélangent.

L'informatique ubiquitaire prend peu à peu une place de plus en plus importante dans notre vie quotidienne.

2. Problématique

Le domaine informatique a ainsi beaucoup évolué. Comme nous l'avons vu, l'explosion d'Internet, l'évolution des réseaux mobiles, des appareils communicants, le développement de l'intelligence artificielle ont accéléré le développement de l'informatique ubiquitaire (systèmes pervasifs).

Ce travail de recherche s'inscrit dans le domaine naissant de l'informatique pervasive dont le thème porte essentiellement sur les enjeux et perspectives de l'informatique pervasive (ubiquitaire).

3. Plan du document

Ce mémoire comporte six chapitres :

- **Chapitre 1 :** ce chapitre comporte l'introduction générale, la problématique et le plan du document.
- **Chapitre 2 :** fait référence aux définitions de l'informatique pervasif, son historique et quelques concepts a partaient a cette discipline.
- **chapitre 3** : il détaille les systèmes informatiques ubiquitaire, Dans un premier temps nous représentons les caractéristiques et le fonctionnement des systèmes informatique pervasifs, puis les moyen de développement de ces derniers, enfin ne décrivons et comparons quelques démarches de l'ingénierie des applications ubiquitaires.
- **Chapitre 4** : présente l'architecture des applications ubiquitaires.
- **Chapitre 5** : présente les différentes technologies sur la quelle repose les systèmes l'informatiques ubiquitaires, les différents domaines qui font appel aux applications ubiquitaires et les équipements utilisés dans l'informatique ubiquitaire;
- **Chapitre 6** : ce chapitre est consacré aux différentes perspectives de l'informatique ubiquitaire et une conclusion générale qui fait la synthèse de ce domaine.

1. Définition

- Définition littéraire :
- ubiquité**:(lat.** ubiquitas, de ubique, partout) faculté d'être présent en plusieurs lieux à la fois selon le dictionnaire Larousse.
- pervasif (mot d'origine anglaise, latin pervasus du participe passé pervardere) aller de toute part, s'insinuer, se propager, se répandre, envahir selon McGraw-Hill Dictionary of Scientific Technical Term.
- Définition existante

❖ La notion de **Pervasive Computing** **«** Informatique omniprésente **»** fut définie initialement par l'article de Mark Weiser, « The Computer for the 21st Century » paru en 1991 dans **Scientific American** (Weiser, 1991) comme étant l'avenir de l'informatique dans lequel l'utilisateur pourra interagir numériquement avec son environnement (intelligent) de façon complètement transparente. L'environnement est assimilé à un ordinateur constitué de plusieurs petits composants qui interagissent en permanence et offrant à l'utilisateur la possibilité d'accéder à des informations et des services. L'informatique diffuse devient donc accessible partout et à tout moment faisant ainsi partie de la vie quotidienne de l'être humain [3] ;

❖ Ziad NEHME a défini l'informatique ubiquitaire comme l'ajout de la capacité de traitement de l'information et la connectivité des éléments. <<L'informatique pervasive, ou ubiquiste, est le terme décrivant le concept d'ajout de capacité de traitement de l'information aux différents objets manufacturés (bâtiment, véhicule, appareil ménager, vêtement...) et leur connexion à un réseau qui permet la communication entre ces différents éléments >> [1] ;

❖ Selon le site wikipidia c'est l'intégration de l'informatique dans la vie quotidienne au niveau de l'interaction homme –machine l'informatique ubiquitaire (ou omniprésente, ou pervasive est le modèle qui suit l'ordinateur de bureau au niveau de l'interaction homme-machine dans lequel le traitement de l'information a été complètement intégré dans tous les objets des activités journalières.

2. Historique

L´informatique ubiquitaire est la troisième ère de l'histoire de l'informatique, qui succède à l'ère des ordinateurs personnels et celle des mainframes caractérisés par la faculté d'un intégration voir la fusion de l'informatique au cœur de nos activités quotidiennes au point d'en disparaître, d'ailleurs certains auteurs lui donne cette définition : Milieu ayant la faculté de percevoir, de raisonner, d'agir et d'interagir afin de fournir des services améliorant la qualité de vie des êtres vivants et notamment des personnes*[4]*.

Le terme ubiquitous computing (en français l'informatique omniprésente) a été inventé la première fois en 1988 par Mark Weiser du Xerox PARC pour désigner sa vision du futur - l'informatique du **xx éme** siècle telle qu'il l'a imaginée. Dans son idée les outils informatiques sont intégrés dans des objets de la vie quotidienne. Les objets sont utilisés aussi bien au travail. Selon lui « les technologies les plus profondes sont celles qui sont devenues invisibles. Celles qui sont nouées ensemble forment le tissu de notre vie quotidienne au point d'en devenir indissociables ». A cette époque nous avons de l'internet et des systèmes de l'intelligence artificielle moins développés, sa vision conçoit une forme de réseau omniprésent.

De puis les années 1990 de nombreuses entreprises commerciales et labo de recherche (Le **Massachusetts Institute of Technology (MIT), Hiroshi Ishii** avec les choses qui pensent (**Things That thinke**) …) axe leurs travaux à cette vision du futur, et exploré le potentiel économique de services omniprésents et d'appareils innovants.

Les améliorations des technologies de télécommunication, en particulier les technologies sans fil tels que le **GSM** accroît l'accessibilité des informations, l'apparition de nouvelles technologies et nouveau matériels Smartphones **PDA** ET **RFID** en accéléré l'évolution de l'informatique pervasive.

a. Evolution technologique :

M. Weiser a proposé d'intégrer des capacités de mobilité et d'intégration des systèmes numériques dans le milieu physique, et ceci de manière spontanée, et à de multiples échelles (micro voir nano objet). Cette vision de l'évolution de l'informatique est conditionnée par des progrès en terme de réduction du coût de production et de performance des capacités techniques (miniaturisation, puissance, faible consommation énergétique, facilite de

communication) et au développement des différentes technologies auxquelles s'articulent plus précisément:

- Réseau de capteurs et les systèmes mobiles.
- Interface Homme-Machine (IHM),
- Intelligence Artificielle (IA),
- l'électronique.

L'évolution de ces différents domaines bénéficie à celui de l'informatique ubiquitaire. Ainsi, l'amélioration des capteurs permet une meilleure connaissance de l'environnement, ce qui constitue l'élément de base aux traitements contextuels des informations (vidéo, radar, audio...), l'évolution des réseaux de communication se construit d'une manière spontanée et leur auto reconnaissance, les interfaces homme-machine deviennent naturelles et intuitives comme la reconnaissance vocale et gestuelle et ajoute la miniaturisation des équipements et supports [5].

S'appuyant sur cette évolution technologique, l'«informatique ubiquitaire» **est entrain de bouleverser la façon d'acquérir et d'utiliser l'information et la connaissance. Son impact est considérable sur la vie quotidienne de l'être humain [5].**

b. Concept de l'informatique pervasive (ubiquitaire)

- **L'informatique pervasif** : capacité de diffusion à travers toutes les parties du système
- **Environnement pervasif** : caractérisé par :

- les objets communicants qui se reconnaissent et se localisent automatiquement entre eux,
- Les objets sont dotés de la capacité de l'interaction spontanée (sans action particulière de l'utilisateur),
- Capacité sans fil,
- Aptitude énergétique importante à fin de garantir un niveau d'autonomie suffisant pour des opérations internes spontanées,

- **Environnement ubiquitaire** :

C'est un ensemble d'éléments qui entourent l'utilisateur, doté de la capacité de communication, de sauvegarde et de calcul (capteur, RIFD) en offrant des services accessibles partout, à tout moment et en même temps.

- **La sensibilité au contexte (context awareness)** : Est une approche prometteuse permettant de supporter les approches ubiquitaires. Pour Dey, un système sensible au contexte est <<un système qui utilise le contexte pour fournir des informations et/ou des services pertinents à un utilisateur. La pertinence dépend de la tâche de l'utilisateur >>, alors que **Henricksen** considère qu'une application est sensible au contexte si elle s'adapte aux contraintes des changements de l'environnement et aux besoins de l'utilisateur [7]
- **Le capteur** : est un dispositif électronique qui peut acquérir une information, calculer et transmettre des données [04].
- **les interfaces d'interaction homme-machine** :

Ziad NEHME considère l'**IHM** comme la capacité de la reconnaissance (voix, vidéo,..), ainsi les interfaces d'interaction homme-machine basées sur des technologies comme la reconnaissance vocale et gestuelle [01].

1. Introduction

Au sein des organisations, les Systèmes d'Information **(SI)** sont directement impactés par l'arrivée de l'informatique pervasive (voir figure 3.1). La mobilité qu'apportent ces nouvelles technologies a étendu les SI bien au-delà des frontières physiques de l'organisation, posant plusieurs défis aux Directions de SI (**DSI)** confrontées à cette évolution. De fait, nous constatons l'émergence d'une nouvelle classe des systèmes d'informations, les systèmes d'information pervasifs **(SIP**). A la différence des SI traditionnels, les SIP s'intègrent progressivement à l'environnement physique, passent à l'arrière-plan, gardent une trace des activités des utilisateurs, analysent les informations et interviennent lorsque cela est nécessaire, afin de mieux répondre aux besoins des utilisateurs (**Kourouthanassis** *et al.* 2006). La conception de cette nouvelle classe des systèmes d'information passe par la compréhension de ces spécificités et des exigences auxquelles ces SI sont soumis.

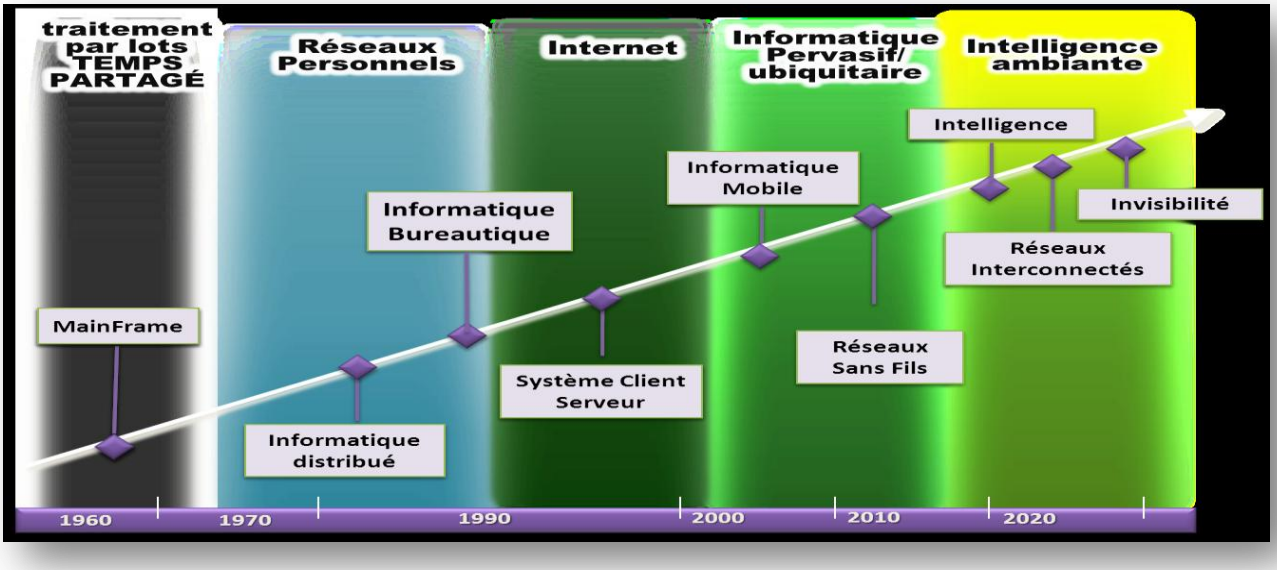


Figure 3.1 Evolution des systèmes d'information pervasifs.

2. Présentation des systèmes d'information pervasifs

La notion des systèmes d'information pervasifs a été proposée au départ comme une vision future de l'informatique par Mark Weiser **[3]** suite à son analyse du marché mondial des ordinateurs et des équipements informatiques (Figure 3.2). Il a remarqué une croissance énorme du nombre d'ordinateurs : un ordinateur pour plusieurs utilisateurs, puis un ordinateur par utilisateur ensuite, plusieurs ordinateurs par utilisateur en utilisant l'intégration des processeurs dans des objets de la vie quotidienne (systèmes embarqués).

Cette nouvelle forme de l'informatique qu'il a nommée informatique ubiquitaire (en anglais **ubiquitous computing**) et dont l'objet viserait à assister implicitement et discrètement un utilisateur dans les tâches qu'il accomplit au quotidien, devenant ainsi la base des systèmes d'information pervasifs **[07]**.

Figure 3.2 Évolution de l'utilisation des ordinateurs

L'évolution technologique actuelle rend aujourd'hui la vision de Mark Weiser réaliste à la suite de l'apparition des nouveaux systèmes embarqués ayant des tailles de plus en plus petites et enfouis dans différents objets de la vie quotidienne. [3]

Les systèmes pervasifs sont un domaine particulier de l'informatique résultant de la convergence des travaux existant dans les domaines de l'informatique mobile et des systèmes distribués. La Figure, donne un aperçu des apports des systèmes distribués, de l'informatique

mobile ainsi que les avancées nécessaires à la réalisation d'un système information pervasif [6].

Les systèmes distribués constituent la rencontre entre les ordinateurs personnels et les réseaux locaux. Ils permettent le partage des capacités et des ressources à travers un réseau et une infrastructure de communication. Cela constitue la première étape de l'informatique pervasive par l'introduction de l'omniprésence de l'information (exemple le Web). [6]

L'informatique mobile est le résultat de l'intégration de la technologie cellulaire et du web pour donner la possibilité aux utilisateurs d'accéder à l'information à n'importe quel endroit et en utilisant différents équipements de petites tailles et de faibles coûts. C'est en quelque sorte la deuxième étape vers l'informatique pervasive « le n'importe où, le n'importe quand. » **[7]**.

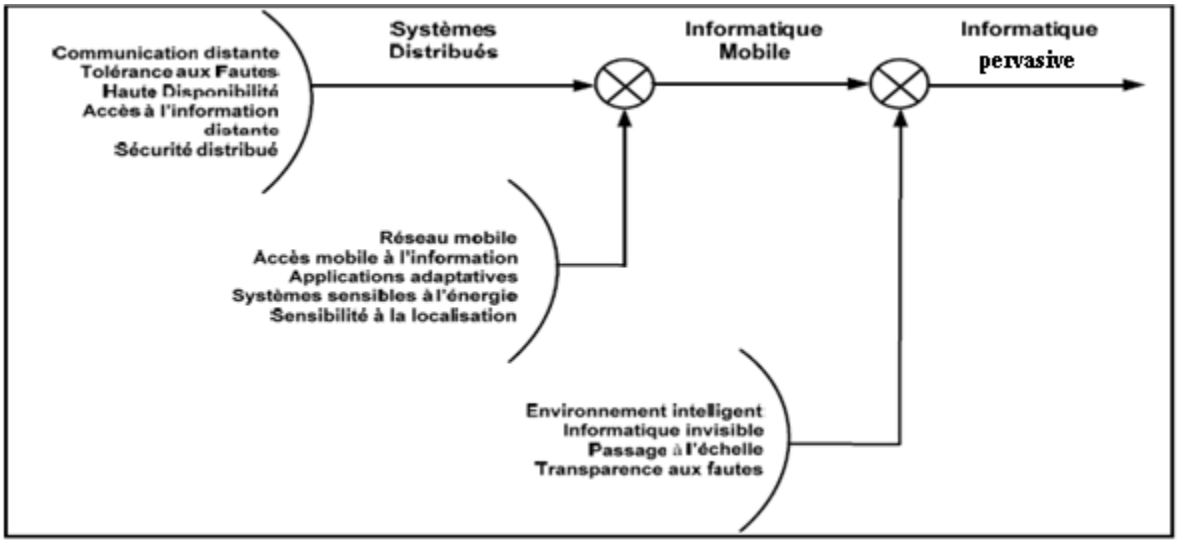


Figure.3.3 l'évolution de l'informatique vers l'informatique pervasive

L'informatique pervasive repose sur la connaissance du contexte de l'usager pour lui délivrer le service approprié au moment opportun. Ainsi, le badge actif (Active Badge) **[08]**, est l'une des premières applications de l'informatique pervasive : elle permet l'accès par identification à certains bureaux du PARC (**Palo Alto Research** Center / Centre de recherches de Xerox) et localise un usager afin de lui transférer ses appels téléphoniques dans le bureau où il se trouve. Depuis, les applications basées sur la géolocalisation se sont multipliées, et le guidage routier par GPS (Global Positioning System / Système de guidage par satellites) en est un exemple.

Le contexte ne peut cependant être circonscrit à la seule localisation d'un usager dans l'espace ; d'une manière plus générale, il doit être envisagé **[09]** comme l'ensemble des informations qui peuvent être utilisées pour caractériser la situation courante d'une entité, tel qu'un individu, un lieu ou un objet considéré comme ayant un rapport avec le service délivré par le système.

My Campus exploite ainsi différentes données issues de l'environnement pour définir le contexte courant de l'usager. Ce système en vigueur sur le campus de Carnegie Mellon se compose de plusieurs "agents" qui remplissent des fonctions spécifiques **[10]**. L'agent "concierge" indique dans quel restaurant universitaire l'étudiant peut prendre ses repas en fonction de ses préférences gastronomiques, de l'heure, de sa localisation sur le campus et des conditions météorologiques. L'agent "réunion " illustre quant à lui, un autre aspect des systèmes pervasifs : celui de l'échange direct de machines à machines ; il assiste les usagers dans le choix d'une date commune de réunion, en parcourant leurs agendas respectifs.

Plus généralement, les applications liées aux systèmes pervasifs relèvent de l'assistance discrète et permanente d'un usager dans l'ensemble de ses activités courantes **[12]** ; en ce sens, elles dépassent le cadre actuel de l'informatique en mobilité.

Pour éviter tout risque de confusion entre les termes utilisés dans le domaine de l'informatique pervasive, nous présentons les définitions suivantes qui le caractérisent **[13]** :

• **ubiquitaire** : Accessible de n'importe où ;

• **mobile** : Qui intègre les terminaux mobiles ;

• **sensible au contexte** : Qui prend en compte le contexte d'exécution ;

• **pervasif** : Qui associe ubiquité, mobilité et sensibilité au contexte ;

• **ambiante** : Qui est intégré dans les objets quotidiens.

La Figure suivante illustre les composants d'un système pervasif [11] :

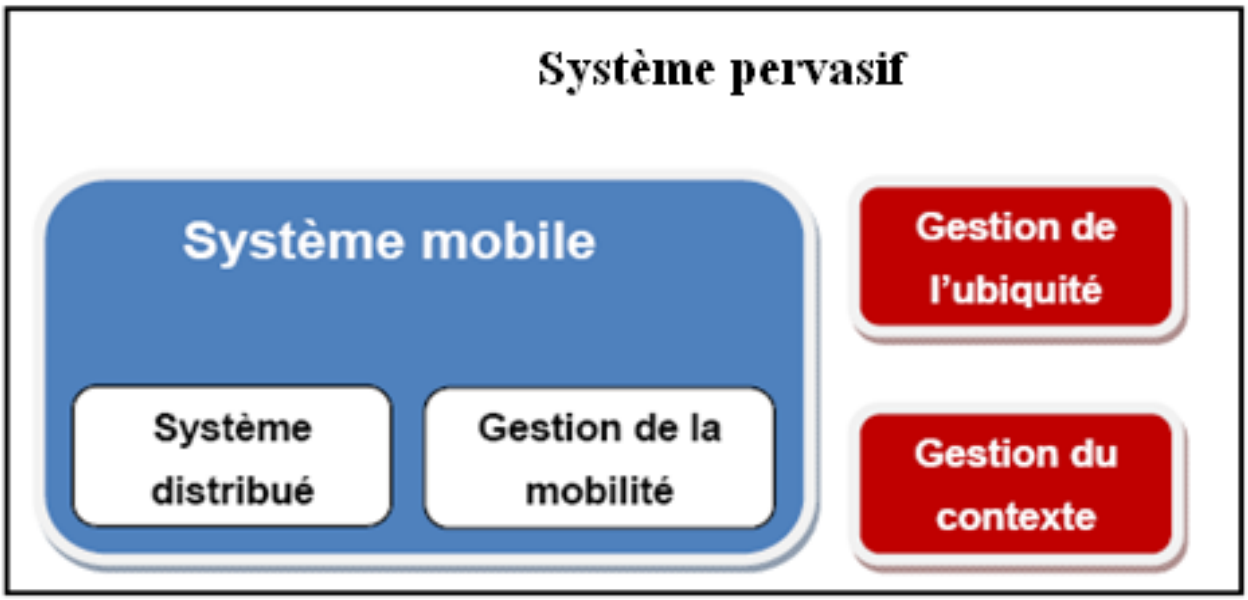


Figure 3.4. Les composants d'un système pervasif. Adaptée de [Saha et Mukherjee, 2003]

3. Les caractéristiques des systèmes informatiques ubiquitaire :

Les SIP sont caractérisés par ***dynamicité,*** **hétérogénéité,** ***sensibilité au contexte*** (environnement physique et utilisateur), *la* ***flexibilité et l'adaptabilité :***

- **Le caractère Dynamique**:

  D'après **Julien Mercadal** le caractère dynamique c'est la disponibilité des objets communicants et leur variance (ajout, suppression) dans le temps. Ainsi, dans un système d'informatique ubiquitaire, la disponibilité des objets communicants peut varier au cours du temps. 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. L'enjeu consiste donc ici à fournir un support de programmation pour découvrir les

objets communicants disponibles dans l'environnement et pour coordonner leurs interactions malgré leur volatilité. [12]

- **Hétérogénéité :**

Les systèmes d'informatique ubiquitaire sont fortement hétérogènes à plusieurs niveaux. Tout d'abord, ils sont déployés dans de nombreux et divers domaines d'application, tels que la domotique [13], la gestion des catastrophes naturelles, l'armée ou encore les soins hospitaliers [14]. La diversité de ces domaines d'application implique toutes sortes d'objets communicants, 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, en passant par le stockage de données. De plus, afin de pouvoir coopérer, ils reposent sur des intergiciels spécifiques (eg : CORBA, DCOM2, Java RMI3, les services Web) supportent différents protocoles réseaux. Ils peuvent également proposer plusieurs modèles d'interaction :

- un vers un ou un vers plusieurs (le modèle événementiel publish/subscribe) ;
- synchrones ou asynchrones ;
- proactifs ou réactifs.

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 encore PNG pour l'encodage des images). Toutefois, l'interopérabilité de toutes ces technologies n'est aujourd'hui pas assurée.

L'enjeu consiste donc ici à abstraire toute la complexité et l'hétérogénéité technologiques afin de faciliter le développement des systèmes d'informatique ubiquitaire.

- **Sensibilité au contexte :**

La notion de sensibilité au contexte concerne l'utilisation du contexte dans les applications. Cette notion est une traduction de l'expression anglaise "context-awareness". Elle caractérise la capacité d'un système à s'adapter aux changements du contexte. Selon Dey et Abowd, « un système est sensible au contexte s'il utilise le contexte pour fournir des informations et des services pertinents pour l'utilisateur, où la pertinence dépend de la tâche demandée par l'utilisateur » [15].

Elle a mis en évidence trois catégories de fonctions liées à la présentation d'information, à l'exécution de services et au stockage d'informations selon le contexte :

- La première catégorie, présentation d'information et services de l'utilisateur,

- La deuxième catégorie, exécution automatique de services.
- la troisième catégorie s'intéressant au stockage d'information selon le contexte.

- **Adaptabilité :**

L'adaptation est, selon le grand dictionnaire l'opération qui consiste à apporter des modifications à un logiciel ou à un système informatique dans le but d'assurer ses fonctions et, si possible, d'améliorer ses performances, dans un environnement d'utilisation" [grand dictionnaire]. Dans le domaine de l'informatique, l'adaptation est étalée sur la notion d'adaptabilité qui caractérise un système qui est capable de changer son comportement lui-même afin d'améliorer ses performances ou de continuer son exécution dans des environnements différents. Dans [Layaïda99], on trouve la définition : « Par adaptabilité, on entend les moyens automatiques ou semi-automatiques qui permettent aux contenus d'être utilisables sur des terminaux ayant des caractéristiques et des ressources très variées. ».

- **La flexibilité** :

Selon l'encyclopédie Larousse, la flexibilité signifie l'aptitude d'un système construit à se plier à une utilisation évolutive ou différente. Il s'agit ici d'un système qui eut être modifié sans être remplacé complètement.

D'après ANSEM Ben CHIKH c'est la capacité de système au changement quelque soit le facteur <<un système est flexible s'il présente une capacité de changement en fonction de n'importe quel facteur et pour toute partie impliquée dans une phase de développement et d'exploitation du système>> [16].

- **Scalabilité:**

Consiste au passage à l'échelle en gérant des volumes de plus en plus grands d'utilisateurs, d'applications et d'appareils connectés. Permet de développer des applications dont le cœur est indépendant du volume, des utilisateurs et des appareils et d'utiliser des techniques d'adaptation pour pouvoir répondre à chaque cas. **[18]**

- **Invisibilité:**

Est la nécessité d'un minimum d'intervention humaine pour s'adapter seul aux changements et l'auto-apprentissage. Par exemple : reconfiguration dynamique des caractéristiques réseaux d'un appareil, accès aux ressources d'un « espace » en fonction de l'appartenance à la zone géographique de cet espace **[18].**

4. Le fonctionnement de systèmes d'informatique ubiquitaire :

Un système d'informatique ubiquitaire permet d'automatiser certaines tâches quotidiennes grâce aux différents objets communicants disponibles.

Etape 1 :

Le système collecte tout d'abord des informations de l'environnement physique

(température ambiante, lumière du soleil, bande passante, présence d'un utilisateur) à partir des objets communicants capables de les capturer. De tels objets communicants sont appelés des capteurs. Ces derniers peuvent être aussi bien physiques que logiciels.

Etape 2 :

Les informations collectées sont ensuite interprétées, filtrées et agrégées par diverses applications, afin de les enrichir de données contextuelles dans le but d'obtenir et de partager des informations de plus haut niveau. À partir de ces dernières, certaines applications peuvent décider des actions à entreprendre (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. Un objet communicant peut être à la fois capteur et actionneur.

Exemple :

Les exemples de systèmes d'informatique ubiquitaire sont nombreux. Nous pouvons citer le traçage de l'ensemble des produits matériels,

- la détection précoce par réseaux de capteurs des accidents écologiques,
- la gestion intégrée des bâtiments (eg : consommation d'énergie et sécurité)
- la surveillance des personnes âgées peu autonomes.
- un système de sécurité pour se protéger contre les intrusions

Cas : un système de sécurité pour se protéger contre les intrusions

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.[22]
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 encolérés télécommunications.

4.1. Les notions principales d'un système d'informatique ubiquitaire

Les systèmes d'informatique ubiquitaire reposent sur deux notions centrales : le contexte à partir duquel ils réagissent et s'adaptent aux différents changements de l'environnement, et la découverte de services pour trouver quels objets communicants sont disponibles dans l'environnement. [23]

- **Contexte**

Il existe de nombreuses définitions, plus ou moins précises, de la notion de contexte en informatique ubiquitaire [20, 09]. Dans ce document, nous adoptons la définition de Dey et al. En considérant le contexte comme «toutes les informations qui peuvent être utilisées pour caractériser la situation d'une entité (i.e., une personne, un lieu ou encore un objet) qui est considérée pertinente pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application eux-mêmes» [24]. Autrement dit, il s'agit de considérer toutes les informations susceptibles d'influencer le comportement du système. Typiquement, ces informations de contexte peuvent inclure des localisations, des identités, des statuts ou encore le temps. Toutefois, la nature du contexte dépend fortement du domaine d'application cible. Par exemple, dans le cas d'un système de détection automatique d'incendie, différentes informations de contexte peuvent être utiles pour caractériser une situation d'incendie, telles que le taux de dioxyde carbone, la température ambiante ou encore la présence ou non de fumée. [25]
Ainsi, à partir de plusieurs pièces d'information de contexte, il est possible d'inférer une nouvelle information de contexte qui pourra à son tour être utilisée par le système. Il apparaît donc important de bien comprendre quelles informations de l'environnement physique constituent le contexte dans un domaine d'application particulière, et de quelle manière elles sont représentées dans le système afin de pouvoir les collectées et les utilisées.

Plusieurs approche ont été proposées pour représenter les informations de contexte d'un environnement d'informatique ubiquitaire, basées sur des ontologies [26, 27], sur des bases de données virtuelles [28] ou encore sur d'autres modèles relationnels [29].

- **Découverte de services**

Les systèmes d'informatique ubiquitaire nécessitent d'accéder aux différentes capacités offertes par les objets communicants disponibles. Pour cela, ils utilisent les mécanismes de découverte de services [30] afin de localiser dynamiquement les objets communicants dont ils ont besoin. La découverte de services permet aux développeurs de ne pas avoir à connaître où sont physiquement localisés les objets communicants. Ainsi, les systèmes ne sont pas statiquement liés à des objets communicants particuliers, ce qui leur permet de s'adapter plus facilement aux changements de l'environnement physique, puisque de nouveaux objets communicants peuvent apparaître et d'autres disparaître. La découverte de services 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 (i.e., les capacités offertes par les objets communicants) sont disponibles dans un environnement, afin de les mettre à disposition des différents systèmes déployés. [31]

5. Enjeux du développement :

Le développement des systèmes d'informatiques ubiquitaire est une tâche difficile, qui requiert à la fois de faire face à l'hétérogénéité des technologies et au caractère dynamique de l'environnement. De plus, ces systèmes sont généralement destinés à évoluer de manière transparente dans un environnement humain, ce qui nécessite de s'intéresser à certaines préoccupations sociales, le cas d'un système d'aide à la personne, si un capteur (eg : de chute) venait à tomber en panne, la sécurité de la personne pourrait être compromise.

De plus, chaque domaine d'application possède des propriétés critiques spécifiques, qu'il est essentiel que le système déployé garantisse. Par exemple, une propriété évidente du domaine de l'aide à la personne est qu'en présence d'une situation anormale, le système alerte automatiquement le personnel médical le plus rapidement possible.

Vérifier un système d'informatique ubiquitaire avant son déploiement en environnement réel est une tâche très compliquée avec les techniques de vérification traditionnelles, notamment à

cause de la taille des systèmes, pouvant impliquer un grand nombre d'objets communicants, du caractère dynamique de l'environnement et de certaines situations difficilement reproductibles (eg : un incendie). L'enjeu consiste donc ici à fournir des garanties de sûreté, soit par construction, soit par analyses, afin de développer et il existe de nombreuses solutions pour développer des systèmes d'informatique ubiquitaire. Ces solutions présentent des approches différentes du problème, reposant sur des intergiciels, des Framework de programmation ou encore des langages dédiés. [33]

6. Intergiciels distribués et Framework de programmation

Une approche classique pour le développement de systèmes d'informatique ubiquitaire est d'utiliser des intergiciels distribués standardisés. Ces derniers facilitent le développement de systèmes distribués traditionnels, en masquant la complexité du réseau et en offrant divers mécanismes génériques (eg : annuaire de services, notification d'événements) pour supporter les interactions entre les différents objets communicants déployés. [33]

Les intergiciels distribués peuvent être regroupés en deux familles :

6.1. Les intergiciels synchrones :

Généralement fondés sur le paradigme des RPC. Parmi les plus utilisés, nous pouvons citer CORBA, DCOM, Java RMI ou encore les services Web [34]. En particulier, ces intergiciels fournissent un moyen pour définir non seulement une représentation abstraite des services offerts par les objets communicants (i.e., leur signature) [33], mais également une représentation intermédiaire des données échangées. Ces définitions sont ensuite utilisées pour générer du support générique de communication et de découverte de services. Elles permettent donc d'abstraire une partie de l'hétérogénéité des systèmes distribués afin d'assurer l'interopérabilité des différents objets communicants. [34]

- **Exemple cobra** :

Définition

Un composant logiciel est une unité de composition dotée d'interfaces spécifiées. Un composant logiciel peut être déployé indépendamment et sujet à une composition par une tierce entité. [36].

➢ Les caractéristiques de CORBA [37] :

- client/serveur,
- orienté objet,

- bus logiciel (ORB),
- RMI,
- Interopérabilité (machines, systèmes, langages, vendeurs)

➢ Les avantages CORBA [38]:

- Elle permette de faire collaborer des applications réparties sur des environnements différents
- Elle diminue la complexité des applications
- elle améliore la qualité logicielle
- elle augmente la productivité

6.2. Les intergiciels asynchrones.

Ils 'utilisent des communications asynchrones, via des échanges de messages ou d'événements. Contrairement aux intergiciels distribués synchrones, souvent proches d'un modèle de programmation client/serveur, les intergiciels distribués asynchrones permettent [39] :

- aux interactions entre les objets communicants de ne pas être liées par l'espace ou le temps.
- les objets communicants sont découplés les uns des autres, favori leur autonomie.

Une famille représentative de tels intergiciels est les MOM8.

Il existe différents paradigmes de communication asynchrone :

- message Queuing,
- publish/subscribe.
- tuple space.

Exemple de paradigmes :

Publish/Subscribe :

Le paradigme Publish/Subscribe définit un mécanisme plus élaboré. Ce mécanisme nécessite au moins un acteur supplémentaire qui hébergera des "sujets" ("topics"). Les serveurs publient leurs événements auprès des sujets pertinents. Les clients s'abonnent aux sujets qui les intéressent. Chaque sujet est responsable de transmettre les événements publiés à la liste des abonnés.

Ce paradigme induit un couplage très lâche entre clients et serveurs. En effet, ces entités ne se connaissent pas si elles n'interagissent que par ce mécanisme. Les clients et serveurs ne connaissent que les sujets. Seuls les sujets ont une connaissance des clients intéressés et Des serveurs d'événements correspondant

Parmi les spécifications du monde des Web Services, WS-Notification est le protocole qui repose sur ce paradigme. Toutefois, seul l'usage de **WS-Eventing** est spécifié par **DPWS**, la norme des Web Services pour les réseaux locaux.

Toutefois, toutes ces solutions, synchrones ou asynchrones, restent peu adaptées pour faire face au caractère très dynamique des environnements d'informatique ubiquitaire. En effet, elles ont été développées pour des réseaux fixes et stables, et ne proposent par conséquent aucune abstraction de programmation pour gérer la mobilité des objets communicants ou encore la notion de contexte **[38].**

6.3. Les nouveaux intergiciels :

Vue le manque de l'adaptabilité des intergiciels synchrones ou asynchrones au caractère très dynamique des environnements d'informatique ubiquitaire, des nouveaux intergiciels sont conçues a fin d'être capables de s'adapter en fonction de leur environnement d'exécution, ont donc été proposés. Ils s'appuient sur les travaux précédents, tout en élevant leur niveau d'abstraction. [39] Parmi ceux-ci, nous pouvons citer :

- Alice,
- Dolmen,
- One. World ,
- LIME ;
- TOTA.

Ces approches fournissent des mécanismes et des abstractions de programmation dédiés pour répondre à certains besoins spécifiques des systèmes d'informatique ubiquitaire.

Exemple : intergiciels dédiés à l'adaptation des applications :

- **One.World**

Le projet **One.World** simplifie la programmation des tâches de découverte de services, de migration ou encore de gestion des erreurs [40]. Les critères fondamentaux de **One .World** [41]:

- les changements se produisant dans l'environnement doivent être explicités afin de permettre aux applications de définir leurs propres stratégies en réaction à ces changements;
- l'environnement d'exécution doit faciliter les compositions dynamiques de services puisque dans un environnement pervasif, les besoins des utilisateurs varient en fonction des services dont ils disposent ;
- la séparation des fonctionnalités et des données manipulées afin de permettre une meilleure gestion et évolution à la fois des services et des données.

L'approche adoptée dans **One.World** est de laisser aux concepteurs et programmeurs des applications ubiquitaires la définition et la programmation des actions à réaliser lorsqu'un changement intervient au sein de l'environnement.

Pour ce faire, **One.World** fournit un cadre générique basé sur la programmation événementielle aussi bien pour la l'assemblage des différentes parties constituant l'application (les composants) que pour la notification des changements aux entités concernées. [41]

Afin de faciliter la programmation des applications, **One.World** propose un ensemble de services dédiés tels que [41] :

- le passage de messages asynchrones (mode de communication par défaut des composants),
- la découverte de services,
- la migration,
- Sur le plan de la communication entre composants distants, one.world prône l'explicitation de la distribution afin de mettre en évidence la nature distribuée des applications ubiquitaires et la dynamicité de leur environnement d'exécution.

7. Langages dédiés distribués :

Plusieurs langages dédiés au développement de systèmes d'informatique ubiquitaire ont été proposés, parmi lesquels nous pouvons citer **AmbientTalk**, **YABS** [44], Habilitation [45] ou encore **VisualRDK** . Ces langages sont fondés sur différents paradigmes de communication (**eg**, le modèle des acteurs, tuple space) et de notation textuelle ou graphique.

Ils offrent également des constructions syntaxiques et des abstractions de haut niveau qui sont proches des concepts du domaine de l'informatique ubiquitaire, et non plus des concepts traditionnels de programmation. Ainsi, de tels langages dédiés permettent de réduire de

manière significative la taille et la complexité des programmes chargés de coordonner les différents objets communicants d'un système.

Toutefois, aucun d'eux ne permet d'élever le niveau d'abstraction de la programmation des systèmes d'informatique ubiquitaire au-delà du code. Par conséquent, il est difficile pour les développeurs d'avoir une vision globale du système et ainsi de pouvoir garantir certaines propriétés du domaine cible.

Les solutions de développement de systèmes d'informatique ubiquitaire sont nombreuses et montrent toutes le besoin de programmer à un plus haut niveau d'abstraction, que ce soit en proposant des mécanismes et des notations dédiés, ou en introduisant des couches de spécification et d'architecture.

Bien que ces solutions permettent de gérer une grande partie de la complexité des systèmes d'informatique ubiquitaire, elles n'exploitent pas suffisamment cette élévation du niveau d'abstraction pour réellement simplifier la programmation des systèmes, notamment en guidant les développeurs et en prenant en compte toutes les spécificités des domaines d'application pour lesquels les systèmes sont développés. Il en va de même pour le processus de vérification. En effet, certaines solutions offrent des mécanismes de tolérance aux fautes pour augmenter la robustesse des systèmes [47,48], d'autres, plus récemment, proposent de formellement spécifier un système d'informatique ubiquitaire afin de pouvoir garantir certaines propriétés [49], mais aucune des solutions existantes ne permet de facilement vérifier un système d'informatique ubiquitaire dans son ensemble[50].

8. Démarches de développement des applications ubiquitaires

Le besoin de concevoir des applications ubiquitaires a engendré l'apparition de modèles et de systèmes dans plusieurs disciplines concernées par des besoins ubiquitaires. Néanmoins, un intérêt moins important a été accordé aux démarches de développement de SI ubiquitaires. Parmi les toutes ces démarches, proposée, nous décrivons d'une manière indépendante les démarches suivante :

- **la Méthode pour l'ingénierie des applications ubiquitaires [Henricksen et Indulska, 2006]**
- **la Démarche dirigée par les modèles pour la création de services ubiquitaires** [Achilleos et al. 2010]
- **La demarche E-CARe (EngineeringContext-Aware and Reactive systems)[Ben Cheikh et al., 2012].**

8.1. Méthode pour l'ingénierie des applications ubiquitaires [Henricksen et Indulska, 2006]

La méthode d'ingénierie proposée dans [51] repose sur un ensemble de concepts et de langages pour la modélisation et la gestion du contexte. Elle utilise, entre autres, une architecture d'une infrastructure logicielle générique à adapter selon les projets. La démarche est illustrée par la figure suivante qui présente un enchaînement d'activités de développement concernant :

- L'analyse,
- la conception, l'implémentation et la programmation.
- la personnalisation de l'infrastructure.
- les tests.

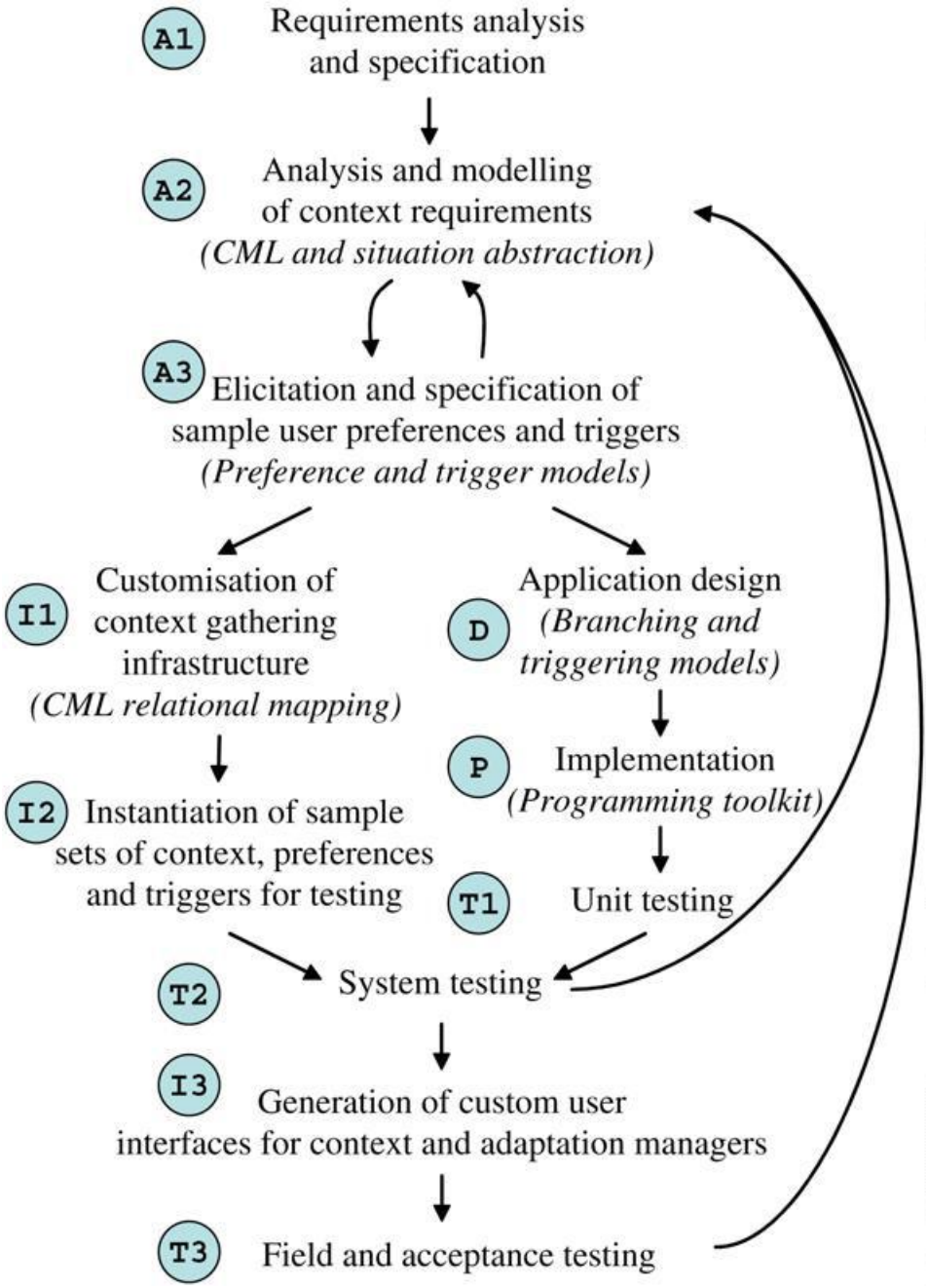


Figure 3.5 Processus de développement d'applications ubiquitaires [Henricksen et Indulska, 2006]

- **Analyse :**

L'analyse permet d'identifier dans un premier temps les exigences et les fonctionnalités de l'application. Ensuite, il est recommandé d'identifier les types d'information contextuelle exigés par ces fonctionnalités en modélisant le contexte avec CML et d'exprimer les situations auxquelles doit répondre le système. Cette étape doit purifier les choix de l'utilisateur et identifier les déclencheurs (évènements) pour les comportements d'adaptation. [51]

- **la conception, l'implémentation et la programmation**

L'analyse est suivie de deux branches qui s'effectuent en parallèle :

La conception et l'implémentation (étape classique de mise en œuvre et codage),

L'infrastructure de gestion du contexte proposée doit être reprise et adaptée au projet pour obtenir une architecture en couches permettant la détection, le traitement et l'usage du contexte. Ensuite l'étape permet de construire des ensembles de données contextuelles, de préférences et d'évènements pour réaliser les tests. [51]

- **Les tests**

La dernière étape, elle permet de passer à l'étape pour construire les interfaces de communication avec les utilisateurs et réaliser, de nouveau, des tests. [51]

Les caractéristiques de cette démarche :

- Elle n'est pas itérative
- elle permet des allers/retours entre les différentes étapes.
- Elle assure ainsi une bonne modélisation du contexte et les mécanismes qui supportent sa gestion.

8.2. Démarche dirigée par les modèles pour la création de services ubiquitaires [Achilleos et al. 2010]

La méthode proposée dans [52] utilise une approche IDM pour fournir un Framework de modélisation supportant un processus de développement de services ubiquitaires.

Le processus de développement est divisé en un ensemble de quatre séquences qui correspondent à une approche IDM classique (comme définie par l'OMG) à savoir :

- la définition d'un langage spécifique au domaine.
- la définition et la validation du modèle de domaine.
- la transformation de modèle à modèle.
- la transformation de modèle à code.

Chacune de ces phases contient un ensemble d'étapes comme illustré dans la figure [52] suivante :

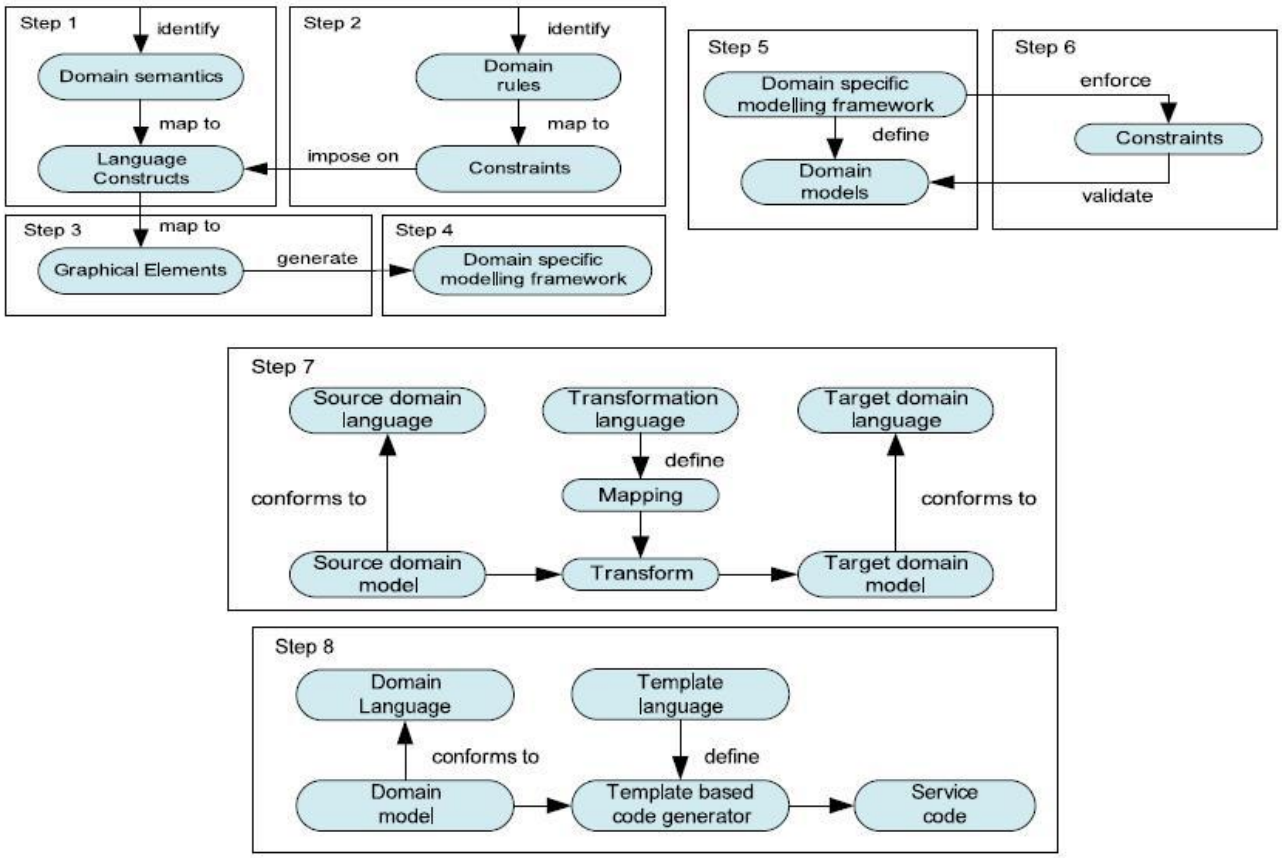


Figure 3.6 La démarche IDM pour la création de services ubiquitaires

[Achilleos et al. 2010]

Phase 1 : Définition d'un langage spécifique au domaine :

Cette phase consiste à définir un langage spécifique au domaine en identifiant et transformant la sémantique du domaine en un ensemble de constructeurs formant un méta modèle (ensemble d'éléments, de relations et de propriétés du domaine). Ce méta modèle doit satisfaire un ensemble de contraintes dérivées des règles du domaine.[54]

Phase 2: Définition et la validation du modèle de domaine**.**

Le méta modèle (de la première phase) est ensuite utilisé dans cette étape afin produire des modèles cohérents de l'application. Cette méthode fournit un Framework générique pour définir les langages du domaine et construire les modèles de l'application. Ce Framework permet de mettre en œuvre les contraintes ce qui garantit la définition de modèles valides (cohérents et complets). [54]

Phase 3 : La transformation de modèle à modèle

Elle consiste à appliquer des transformations modèles à modèles pour transformer les modèles obtenus avec le langage spécifique au domaine en des modèles spécifiques à la plateforme. [54]

Phase 4: La transformation de modèle à code

Cette phase concernant des transformations modèle à code permet de générer le code de l'application à partir des modèles spécifiques à la plateforme. Il s'agit de la phase responsable de l'implémentation [54].

8.3. La demarche E-CARe (Engineering Context-Aware and Reactive systems) [Ben Cheikh et al., 2012].

Présentation de la démarche

La démarche E-CARe se base sur un ensemble de principes considérés comme des choix facilitant la séparation des tâches des développeurs et la spécification des modèles ubiquitaires. Tout d'abord, nous séparons les phases de la démarche en des phases fonctionnelles, ubiquitaires et techniques.

D'ailleurs, la séparation des besoins fonctionnels et techniques a montré son intérêt dans les démarches traditionnelles de conception de SI telles que 2TUP et Symphony. La création de phases d'analyse et de conception ubiquitaires induit l'intervention d'un nouvel acteur : l'analyste-concepteur de contexte. Ces choix ont motivé la création d'un cycle de vie de la démarche contenant une branche de développement ubiquitaire indépendante des besoins techniques et fonctionnels et qui fait l'objet de cette démarche [55].

- Séparation des besoins ubiquitaires, fonctionnels et techniques
- Tout projet de développement logiciel est susceptible de subir des modifications et des changements continuels imposés par les clients ou par l'environnement (changement des politiques, des technologies ou des stratégies). C'est dans ce cadre, que le développement logiciel doit être réutilisable et les modèles fournis doivent être faiblement couplés pour faciliter les modifications et les mises à jour sans affecter tout le système. La séparation des besoins selon leur nature (fonctionnelle, ubiquitaire ou technique) est la solution adoptée dans notre démarche.
- Séparation des besoins fonctionnels et techniques

- Les besoins fonctionnels résument l'ensemble des métiers des utilisateurs. Il s'agit d'identifier ce que réalisent le système, les services fournis et les interactions nécessaires avec les utilisateurs indépendamment des technologies et des équipements utilisés.
- Les besoins techniques regroupent l'ensemble des choix matériels et logiciels et leur intégration dans l'environnement existant de l'entreprise. Il est nécessaire de construire une architecture technique indépendante des besoins fonctionnels et utilisée par la suite pour le prototypage du système.
- Un SI subit généralement des contraintes fonctionnelles et techniques séparées qui peuvent être traitées de façon parallèle sans affecter la totalité du système. Des approches comme **2TUP** [66] et Symphony [**Hassine, 2005**] considèrent cette séparation en adoptant des démarches de développement avec un cycle de vie en **Y** (voir la figure suivante).

La séparation de ces besoins présente les avantages suivants :

- La réutilisation. Les composants fonctionnels peuvent être utilisés avec différentes architectures techniques correspondant aux différentes contraintes matérielles et logicielles imposées.
- De même, les composants techniques conçus peuvent être réutilisés avec des composants fonctionnels différents.
- La flexibilité. Les systèmes ainsi conçus se basent sur le principe de découplage entre composants, ce qui facilite l'évolution de chaque composant indépendamment des autres. Ces systèmes acceptent plus facilement les innovations et la créativité des acteurs impliqués dans le projet.
- Le gain de temps. Les tâches de spécifications fonctionnelle et technique menées en parallèle permettent de gagner du temps dans les délais du projet puisque les développeurs de disciplines différentes peuvent avancer indépendamment. L'évolution du travail est plus fluide et aucun acteur n'est dépendant de l'autre.

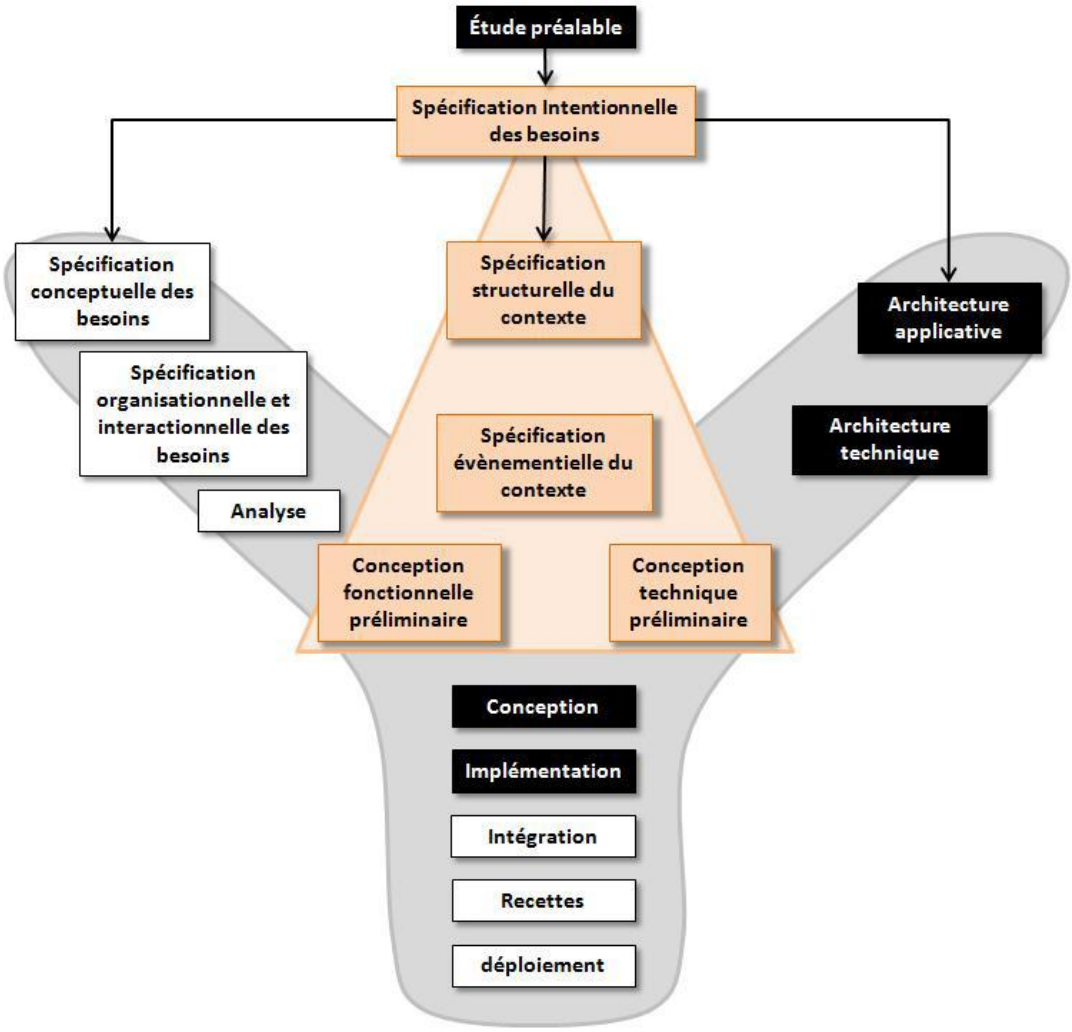


Figure3.7 le cycle de vie Ecar

Spécificités des besoins ubiquitaires En plus des besoins fonctionnels et techniques, les systèmes ubiquitaires comportent des besoins liés à l'ubiquité, la mobilité et la distribution qui permettent d'utiliser des technologies innovantes pour créer des nouvelles fonctionnalités proches de l'utilisateur et de ses attentes. La spécification et l'analyse de ces besoins est loin d'être une tâche triviale et facile puisqu'elle fait intervenir des données personnelles des utilisateurs délicates à manipuler et à utiliser. Ainsi, nous séparons la spécification des besoins ubiquitaires des autres besoins fonctionnels et techniques et proposons une démarche indépendante pour l'identification et l'analyse de ces besoins. Cette séparation est due aux raisons suivantes :

- Les besoins ubiquitaires représentent en général des paramètres ajoutés aux métiers pour déterminer comment une fonctionnalité est réalisée : livrer des

résultats, trier des résultats ou imposer des modes d'interaction. Dans ce cas les besoins ubiquitaires ne s'intéressent pas au cœur métier, mais plutôt à la manière de le présenter aux utilisateurs.

- D'autres besoins ubiquitaires font intervenir le cœur métier pour définir différentes manières d'organiser ces fonctionnalités en vue de les adapter à l'utilisateur final. La partie concernant la spécification de ces besoins ubiquitaires peut être traitée séparément du résultat d'adaptation métier qui est étudié dans la démarche purement fonctionnelle. Cette séparation est nécessaire dans la mesure où les données paramétrant un besoin d'adaptation métier permettent de construire le contexte.
- Les données contextuelles qui constituent la base de tous les besoins ubiquitaires sont différentes des données systèmes et n'affectent pas les métiers de l'entreprise.
- Les besoins ubiquitaires liés à la distribution et la mobilité permettent de définir des choix matériels et architecturaux qui sont traités avec le reste des besoins techniques et non dans la branche ubiquitaire dédiée à la sensibilité et la réaction au contexte.
- Les technologies ubiquitaires constituent des besoins traités dans la branche technique et qui n'affectent pas la spécification des besoins contextuels et des fonctionnalités qui en découlent.
- Cette vision des besoins ubiquitaires incite à créer une démarche de spécification des besoins de sensibilité et de réaction au contexte qui étudie de façon séparée les différentes possibilités pour bénéficier des données contextuelles afin de fournir des fonctionnalités et des services plus proches de l'utilisateur et de son mode de pensée. Les aspects ubiquitaires concernant la distribution et la mobilité sont naturellement traités dans les spécifications techniques. [55]

– Le **tableau 1.1** résume les objectifs des trois dimensions menées en parallèle à savoir les spécifications : fonctionnelles, ubiquitaires et techniques.

Les spécifications fonctionnelles	Les spécifications ubiquitaires	Les spécifications techniques
➢ Définir l'organisation métier ➢ Définir les modèles d'interaction ➢ Définir les données métier ➢ Modéliser les processus métier	Identifier les besoins d'adaptation et d'accès _ Définir les situations critiques ou d'alerte _ Modéliser les données contextuelles _ Définir l'acquisition et l'usage du contexte	❖ Définir les besoins matériels prenant en compte la distribution et la mobilité ❖ Définir les besoins de sécurité ❖ Définir les choix logiciels,(y compris les technologies Ubiquitaires)

Tableau 1 : les différentes spécifications

- **L'analyste-concepteur de contexte**

Le rôle de concepteur de contexte a souvent été lié au développement de jeux vidéo pour désigner des acteurs responsables de la définition du décor et des arrière-plans. Par ailleurs, ce rôle a été mentionné dans des travaux portant sur les systèmes ubiquitaires pour désigner la personne responsable de la modélisation du contexte [56]. Pourtant cet usage n'a pas été justifié et les responsabilités de ce rôle n'ont pas été délimitées. Dans notre approche nous proposons de mettre en évidence ce rôle qui prend en charge toutes les phases de la branche ubiquitaire E-CARe. [55]

L'analyste-concepteur de contexte est responsable de l'identification des fonctionnalités ubiquitaires supportées par l'application. Ces fonctionnalités font intervenir des données de contexte propres aux utilisateurs ou à l'environnement de l'application. Ainsi, l'analyste-concepteur de contexte doit être capable de définir ces données contextuelles, leurs origines et leurs usages, de les modéliser et de les préparer à être exploitées par les développeurs de

l'application. Par conséquent, l'analyste-concepteur de contexte doit être doté des compétences énumérées ci-dessous :

_ Expertise du domaine. L'analyste-concepteur de contexte doit connaître le domaine de l'application pour pouvoir définir des fonctionnalités ubiquitaires respectant les différentes entités du domaine et leurs liens entre elles. Une bonne connaissance du domaine permet de créer des fonctionnalités originales et innovantes proches de l'utilisateur.

_ Expertise des besoins de l'utilisateur. L'analyste-concepteur de contexte doit être capable de définir toutes les catégories d'utilisateurs et les besoins de chacune de ces catégories. En effet, les utilisateurs selon leurs âges, leurs niveaux professionnels, leurs personnalités, etc …, ne présentent pas les mêmes besoins et attentes d'une application. Ainsi, l'analyste-concepteur de contexte doit être attentif à ces aspects et doit se procurer des mécanismes pour mieux identifier ces besoins et les utiliser pour définir les profils des utilisateurs.

_ Expertise de l'environnement. L'analyste-concepteur de contexte doit être capable d'identifier les différentes données de l'environnement qui peuvent influencer certaines fonctionnalités de l'application et les utiliser pour définir le contexte de l'application. Cette expertise doit permettre de plus de définir comment exploiter l'environnement pour dériver des données de contexte ce qui représente les mécanismes d'acquisition du contexte.

_ Connaissance des technologies ubiquitaires. La définition des données contextuelles et des mécanismes d'acquisition requiert une bonne connaissance des technologies ubiquitaires et de la possibilité de les utiliser pour ces finalités. Ainsi, la créativité de l'analyste-concepteur de contexte est limitée par les technologies ubiquitaires existantes.

_ Expertise informatique. L'analyste-concepteur de contexte participe au développement de l'application et doit être doté des compétences informatiques pour la modélisation et la spécification des besoins. Ainsi, il doit être expert des langages de modélisation pour fournir des modèles formels pouvant être exploités par la suite. [54]

8.4. Les comparaisons entre les différentes méthodes :

Dans cette dernière partie du chapitre, nous avons présenté différents travaux proposant des démarches de développement de systèmes ubiquitaires. Nous comparons maintenant ces démarches a base des critères énumérés en dessous (voir tableau)[55] :

- Le champ d'application
- Les concepts de bases
- la complétude
- La dynamicité
- L'ingénierie des besoins
- Le guidage de la modélisation du contexte

Les démarches[Henricksen et Indulska, 2006] et [Achilleos et al., 2010] sont des démarches incomplètes car elles traitent pas tous les caractères de l'ubiquité (la mobilité et la disponibilité ,la dynamicité) mais elle se focalisent sur la sensibilité au contexte comme caractère de base pas un intérêt particulier à la sensibilité au contexte considérée comme caractère des systèmes ubiquitaires. Cependant, la méthode au ECAR [benchikh 2012] couvre toutes les spécifications (technique, fonctionnel et ubiquitaire) et elle accorde un intérêt majeur pour l'analyse et conception, donc on peut dire que cette démarche est conçue afin de construire des systèmes ubiquitaire complet.

Démarché	Champs d'application	Concepts de base	Complétude	Dynamicité	Ingénierie des besoins	Modélisation guidée
Achilleos et al. 2010]	Services ubiquitaires	Métamodèle de contexte	non	Non	non	non
Henricksen et Indulska, 2006]	Systèmes sensibles au contexte	_Architecture générique _ Langage CML _ Modèle de préférences _ Modèle de règles	oui	Oui	non	non
ecar	Service et besoin ubiquitaire	Architecture générique _ basé sur le fremowrk ECAR _ spécification technique, fonctionnelle et ubiquitaire	oui	Oui	oui	oui

Tableau 2 : résume des comparaisons entre les démarches

9. Conclusion :

La complexité des systèmes pervasifs a poussé à l'invention d'un certain nombre de démarches plus spécifiques qui sont apparues durant les dernières années supportant le développement des applications ubiquitaires. Malgré, que les premières sont marquées par des insuffisances sur le plan de la satisfaction des caractères ubiquitaires et les spécifications fonctionnels et techniques, cependant le développement de la démarche E-CARe

(**Engineering Context-Aware and Reactive systems**) est considérée comme la démarche la plus performante à ce moment dans la construction des systèmes ubiquitaires.

1. Introduction :

L'architecture logicielle facilite la compréhension de systèmes complexes en les représentant à un haut niveau d'abstraction et en masquant certains détails technologiques. Ainsi, les descriptions architecturales ont rapidement et naturellement constitué un formidable vecteur de communication entre les différents intervenants d'un projet informatique. Aujourd'hui, la maîtrise des architectures logicielles offre de nombreux autres avantages dans le développement logiciel, notamment en termes de réutilisation, de vérification et d'évolution. En effet, l'architecture logicielle permet une meilleure séparation des préoccupations, ainsi que de nouvelles opportunités d'analyse. Elle permet également de factoriser certaines expériences passées et éprouvées sous la forme d'un style architectural.

2. Définition et caractéristiques:

Il existe de nombreuses définitions d'une architecture logicielle mais chacune d'elles met l'accent sur certains aspects particuliers de l'architecture logicielle. L'architecture d'une application ou d'un système logiciel définit ce dernier en termes de composants et d'interactions entre ces composants. <<**Une architecture est une infrastructure composée de modules actifs, d'un mécanisme d'interaction entre ces modules et d'un ensemble de règles qui gouvernent cette interaction**>> [57].

3. **Description de l'architecture logicielle** :

Cette architecture définie dans [58] consiste à:

• Décrire l'organisation générale d'un système et sa décomposition en sous-systèmes ou composants

• Déterminer les interfaces entre les sous-systèmes

• Décrire les interactions et le flot de contrôle entre les sous systèmes

• Décrire également les composants utilisés pour implanter les fonctionnalités des sous-systèmes.

4. Utilité d'une architecture logicielle :

La conception d'une architecture logicielle est d'une importance capitale pour la réussite d'un projet informatique car elle :

- facilite la compréhension des grands systèmes complexes en donnant une vue de haut-niveau de leur structure et de leurs contraintes.

- favorise l'identification des éléments réutilisables, parties de conception, composants, caractéristiques, fonctions ou données communes
- fournit un plan de haut-niveau du développement et de l'intégration des modules en mettant en évidence les composants, les interactions et les dépendances.
- offre la capacité de modifier et de maintenir en production une application sur une période de vie assez longue. Une architecture bien conçue doit aussi être maintenable.
- offre une base pour l'analyse plus approfondie de la conception du logiciel.

A l'ère de l'informatique ubiquitaire, la maîtrise des architectures logicielles offre de nombreux autres avantages dans le développement logiciel, notamment en termes de réutilisation, de vérification et d'évolution. En effet, l'architecture logicielle permet une meilleure séparation des préoccupations, ainsi que de nouvelles opportunités d'analyse

Les éléments architecturaux sont les composants, les interfaces composantes, dépendances entre composants et connecteurs.

5. Les modèles architecturaux :

La diversité des plates-formes et des dispositifs utilisés a imposé aussi une grande évolution (développement) dans le monde des systèmes interactifs (interfaces graphiques, ubiquitaire). Ce monde est riche, il comprend des résultats concernant les méthodologies, les techniques, les modèles, les plates-formes et les langages. La bonne manipulation de tous ces éléments aide à construire des Interfaces Homme-Machine «intelligentes» faciles à utiliser.

Dans le domaine de l'informatique ubiquitaire, le concept d'architecture décrit la façon dont les composants logiciels et matériels sont organisés et assemblés pour réaliser un système interactif. Il existe plusieurs modèles d'architecture spécialisés, par exemple : Arch, **MVC, PAC, PAC AMODE, AMF** etc. Tous ces modèles ont comme base une séparation entre le Noyau Fonctionnel et la Présentation (l'interface utilisateur).

Dans cette partie nous faisons une courte présentation des quelques modèles généraux d'architecture existants, qui sont des modèles de référence : le modèle **multi-agents PAC.**

- **Le modèle multi-agents PAC**

La gestion dynamique de la présentation nécessite une conception modulaire du code pour faciliter le changement d'objet d'interaction en cours d'exécution. Il existe plusieurs modèles utiles à la construction de logiciels interactifs, mais les plus intéressants sont de type multi-

agents. Ces modèles offrent en effet une forte modularité et une ouverture vers les traitements physiquement distribués.

Le modèle PAC (Présentation Abstraction Contrôleur) est un modèle architectural multi-agent. Il propose une conception de l'architecture d'un logiciel interactif, fondée sur une structure arborescente d'agents. Chaque agent PAC se décompose en trois facettes:

- la facette présentation définit la représentation externe de l'agent en entrée comme en sortie. Un agent PAC peut comporter aucune ou plusieurs facettes présentations.
- la facette abstraction définit le Noyau Fonctionnel de l'agent (ses compétences de calcul). Un agent PAC ne peut comporter aucune ou une seule facette abstraction.
- la facette contrôle maintient la cohérence entre la présentation et l'abstraction de même que la communication avec les agents voisins de la hiérarchie PAC. Un agent PAC doit comporter une et unique facette contrôle. [60]

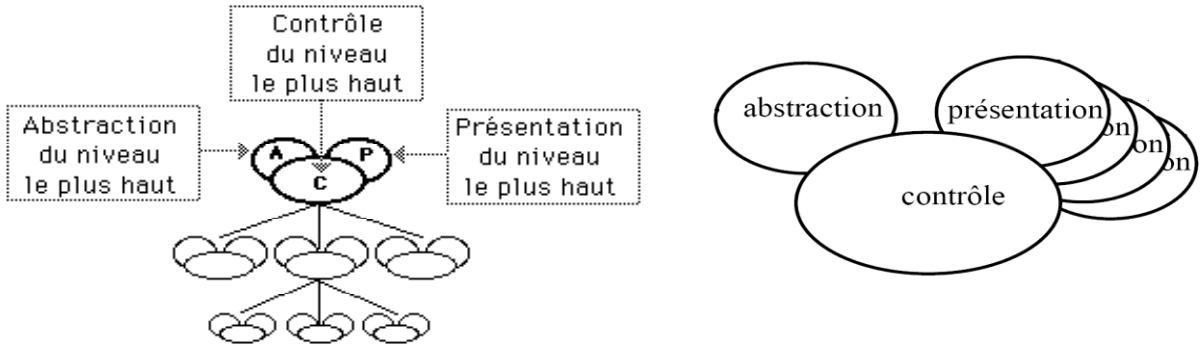


Figure 4.1. La représentation du modèle PAC

6. **Langages de description d'architectures logicielles :**

Arthur Clarke a défini les MIL comme des langages de haut niveau indépendants des plateformes << Les MIL sont des langages de haut niveau dans lesquels on décrit l'interconnexion des composants, indépendamment de la plate-forme de déploiement. On sépare alors la phase de programmation de composants logiciels pris individuellement, de la phase de configuration d'une application qui consiste à assembler ces mêmes composants selon un schéma d'interaction propre à l'application. Un environnement d'exécution est utilisé pour encapsuler les communications entre les modules et réaliser l'adaptation des données>> [61].

7. Les langages de définition d'architecture logicielle (ADL) :

Selon **J. Ferber,** Les ADL sont des identifiions des styles d'architecture logicielles, généralement les contraintes topologiques << Les langages de définition d'architecture **l**ogicielle **(Architectural Description Language ou ADL**) identifient différents styles d'architectures logicielles. Ce sont en général des contraintes topologiques qui définissent les composants logiciels utilisables et les interactions entre ceux-ci. Une architecture est alors décrite par des composants (aussi appelés des tâches ou des modules) reliés logiquement >> [62].

8. Défis liés aux architectures logicielles

Certains auteurs considèrent que l'architecture logicielle est aujourd'hui dans une période bien évoluée. Néanmoins, si l'importance de l'architecture est désormais bien reconnue, la communauté s'accorde sur le fait que des problèmes majeurs et complexe sont encore à traiter dans ce domaine. Attardons-nous sur trois défis majeurs :

– La description des architectures,

– L'évaluation des architectures,

– La conception des architectures.

Les deux premiers points ont donné lieu à de nombreuses recherches sur les ADL, c'est à dire les Langages de Description d'Architecture. Un ADL peut être informel (utilisation de schémas par exemple), semi-formel (en UML par exemple) ou formel. Un langage de description d'architecture fournit les éléments pour représenter une ou plusieurs vues architecturale se focalisant sur une préoccupation particulière. De plus, la description d'une architecture à l'aide d'un ADL peut être compréhensible par une machine, ou transformable en un format compréhensible. Les ADL ont pour but de favoriser la compréhension des architectures, leur dissémination mais aussi leur évaluation. Une description peut ainsi être analysée et utilisée afin d'automatiser l'implémentation de certains aspects.

Par ailleurs, chaque langage de description d'architecture se distingue par ses capacités de modélisation qui proviennent directement des objectifs poursuivis par cet ADL [63]. Par exemple, **Aesop** [64] spécifie les architectures d'applications en se basant sur un ensemble de styles architecturaux. SADL [64] est un langage de description d'architecture structural. Il se concentre sur le raffinement des architectures du système sur différentes couches d'abstraction de façon formelle. Cela permet la traçabilité des variantes et des évolutions du système. **UniCon** [65] permet de générer du code de liaison afin de construire et vérifier les architectures à partir d'éléments architecturaux prédéfinis utilisant les protocoles communs.

MetaH [66] se concentre sur la conception, la vérification et la validation de l'assemblage et du déploiement de systèmes temps réel critique. Il est utilisé pour modéliser les architectures en particulier dans le domaine de la navigation et du contrôle.

Certains langages de description d'architecture permettent de décrire des interactions collaboratives entre les éléments structuraux du système à l'exécution. En d'autres termes, ils permettent de modéliser, simuler et analyser le comportement dynamique d'un système. C2, Darwin, Rapide et Wright constituent des exemples de tels **ADLs**. C2 [67] se focalise sur la description des architectures d'application réparties et dynamiques. Darwin [68] traite de la configuration et de l'instanciation des systèmes distribués et dynamique de façon formelle. Rapide [69] est un langage permettant de simuler le comportement dynamique à l'aide de **POSER** (partial ordered event) 2. Il permet de vérifier des propriétés telles que la synchronisation et la concurrence sur des assemblages de composants. Comme **Rapide, Wright** [70] est utilisé pour modéliser et analyser formellement le comportement dynamique des systèmes concurrents. Mais, **Wright** se concentre plus sur la vérification de conformité de l'assemblage des éléments architecturaux et la détection d'inter-blocages des systèmes concurrents au cours de l'exécution. Il emploie le concept de **CSP** (***Communicating Sequential Process***) [71] pour décrire l'architecture du système logiciel.

Un constat s'impose par rapport à ces travaux sur les A**DL** : ils n'ont pas percés et ne sont pas utilisés du tout au delà de quelques laboratoires. De nombreuses avancées sont encore nécessaires pour une modélisation effective des architectures. Le troisième défi mentionné ici est lié à l'activité de conception des architectures logicielles. La tâche de conception reste en effet très complexe et soulève de nombreux problèmes. Il est en fait très urgent de faire émerger des solutions pragmatiques et ***scalables*** (capables de passer à l'échelle) facilitant le travail des architectes (et concepteurs au sens large).

Un architecte doit produire une architecture répondant aux exigences du projet considéré tout en assurant un ensemble de qualités logicielles fondamentales. En particulier, une architecture doit respecter les propriétés suivantes :

– Abstraction, par nature, une architecture est une abstraction. Elle définit un ensemble de composants au travers d'interfaces, de propriétés et de documentations diverses. Ces éléments représentent une enveloppe, un cahier des charges qui doit ensuite être implanté. Il en va de même pour les connexions qui sont définies de façon abstraite. L'abstraction est un élément fondamental : il donne la latitude nécessaire aux concepteurs/développeurs d'implanter un système dans le langage cible et sur les plates-formes cibles tout en disposant d'un canevas de travail [72].

– Modularité, la modularité d'une architecture repose sur la notion de composant. Cette notion, si correctement appliquée, permet de mettre en œuvre le principe d'encapsulation et de séparer les notions de traitement et de communication. Les composants néanmoins visent la décomposition structurelle et ne traitent pas de façon explicite les autres types de décompositions. **Parnas** [74] a indiqué il y a déjà longtemps qu'un logiciel est caractérisé par de nombreuses dimensions (dont certaines non fonctionnelles comme la sécurité par exemple). La prise en compte des différentes propriétés non fonctionnelles peut rapidement alourdir les composants au travers de la définition de nombreuses interfaces difficiles à gérer. Une bonne modularité est donc très dépendante du problème abordé et difficilement réutilisable : idéalement, elle doit répondre uniquement au problème considéré !

– Cohérence, ceci nous amène de façon naturelle vers la notion de cohérence. Un défi majeur de toute conception est de créer des artefacts cohérents, regroupant des concepts sémantiquement proches dans le contexte considéré. Ceci permet de limiter les communications entre composants et de favoriser l'évolution en limitant les effets de bord lors de modifications. Ici aussi, la cohérence des artéfacts dépend fortement du problème abordé et est difficilement réutilisable.[73]

– Couplage, le couplage est complémentaire à la cohérence. Il caractérise la force des liens entre les artefacts logiciels, les composants dans le cas d'une architecture logicielle, ainsi que la nature des liens et leurs nombre. Il est impératif de maîtriser le nombre et la complexité de ces liens. Ceci permet une meilleure lisibilité, ou compréhension, du code et favorise l'évolution. Des liens de couplage bien maîtrisés limitent la propagation des modifications [72]

– Evolutivité, un défi majeur pour tous les architectes est l'évolutivité des logiciels. Bien sûr, il n'est pas possible de structurer un système de façon à prévoir et préparer tous les changements possibles. Mais, il est également acquis que de nombreux systèmes actuels sont amenés à s'adapter fréquemment à de nouveaux contextes ou de nouveaux besoins utilisateur. Dans certains cas, ces évolutions doivent se faire de façon dynamique, c'est-à-dire lors de l'exécution d'un système logiciel. Dans ces cas là, des mécanismes permettant l'évolution sont nécessaires [74].

9. **Styles architecturaux** :

Un aspect important des architectures logicielles est le concept de style architectural [97].Ce dernier caractérise une famille de systèmes qui utilise les même types de composants, d'interactions, de contraintes structurelles et sémantiques, et d'analyses. Parmi les styles

architecturaux les plus répandus, nous pouvons citer le style **pipes and filters,** le style **client/-serveur** ou encore le style architecture **en couches** [97].

Le but premier des styles architecturaux est de faciliter la conception architecturale ainsi que sa réutilisation. En effet, un style architectural capture et exploite certains motifs et langages d'un domaine d'application particulier. De plus, en contraignant l'espace de conception, un style architectural permet des analyses spécifiques et puissantes qui vont au-delà de ce qu'il est possible de faire a vec des éléments architecturaux généralistes [75].

- **Exemple des styles architecturaux utilise dans les applications pervasifs :**

– **L'architecture client/serveur :**

Elle s'articule en général autour d'un réseau. Deux types d'ordinateurs sont interconnectés au réseau. Le serveur assure la gestion des données partagées entre les utilisateurs. Le client gère l'interface de la station de travail d'un utilisateur. Les deux communiquent par des protocoles et les programmes applicatifs sont sur le client et/ou le serveur.

Client/Médiateur/Serveur) :

– le client : processus demandant l'exécution d'une opération a un autre processus serveur par l'envoi d'un message contenant le descriptif de l'opération a exécuter et attendant une réponse `a cette opération par un message en retour.

– le serveur : processus accomplissant une opération sur demande d'un client et transmettant la réponse à ce client.

– le middleware : ensemble des services logiciels construits au-dessus d'un protocole de transport afin de permettre l'´echange des requêtes et de réponses associées entre client et serveur de manière transparente. Les différents modèles de client-serveur : le client- serveur de donnes, le client -serveur de présentation et le client -serveur de traitement.

De même, il existe différentes architectures: l'architecture 2-tiers, l'architecture 3-tiers et l'architecture multi-tiers.

Exemple : L'architecture multi-tiers (à base de composants)

Présentation :

Dans l'architecture 3-tiers, pour répondre à une demande de service émise par un client, un serveur peut utiliser les services d'un ou plusieurs autres serveurs afin de fournir son propre service (d'o`u le multi-tiers), le niveau médian de la plupart des applications 3-tiers n'est pas implémenté sous forme d'un programme monolithique, mais comme un ensemble de composants utilisés lors de diverses transactions initiées par les clients. Chaque composant automatise une fonction relativement limitée. Les clients combinent souvent plusieurs

composants du niveau médian en une transaction opérationnelle unique. Un composant peut appeler d'autres composants pour implémenter une requête. Certains composants peuvent en outre se comporter comme des passerelles encapsulant des applications. La plupart du temps, les 3-tiers sont donc en réalité N-tiers [42).

Avantages du client/serveur N-tiers

- D´développement de grandes applications par étapes :
- Réutilisation des composants.
- Accès fiable des clients aux données.
- Durabilités et bonification des applications.
- Inclusion de composants standards dans des applications sur mesure

10. Architecture logique

L'architecture logique occupe une place centrale. Instrument privilégié pour la construction et la maintenance du système, pivot sur lequel s'articulent le métier et sa traduction logicielle, elle constitue la référence pour l'organisation des projets, la construction technique et le plan de progression. Toutefois, l'architecture logique n'est pas un modèle abstrait, ni une cartographie fonctionnelle cible lointaine. Il s'agit plus pragmatiquement de la description des constituants du système et de leurs relations.

– E**xemple des architectures logiques**

- **Architecture dirigée par les évènements :**

L'architecture dirigée par les évènements (EDA) est le cadre exécutable englobant toute approche orientée évènements. Le principe de cette architecture est que chaque évènement interne ou externe, quand il se produit, est envoyé immédiatement à toutes les parties intéressées (humaines ou automatisées). Par nature, une architecture EDA est faiblement couplée et très distribuée. Le producteur d'un évènement ne connait pas les parties intéressées ni les traitements que va subir l'évènement. Ainsi, cette architecture est le choix idéal pour des flux d'informations et des tâches asynchrones. Mais, cette architecture rencontre encore certains verrous reliés principalement à l'usage d'un standard permettant l'interopérabilité entre les composants hétérogènes [76].

Infrastructure de l'architecture

Une architecture dirigée par les évènements est basée sur un ensemble de composants qui constitue ce qu'on appelle une infrastructure CEP. Cette infrastructure est responsable de l'interfaçage avec les couches techniques du système et fournit une plateforme pour l'implémentation du CEP. Un exemple d'architecture fournie par [75] est présenté dans la (Figure). Les composants de cette architecture sont décrits ci-dessous.

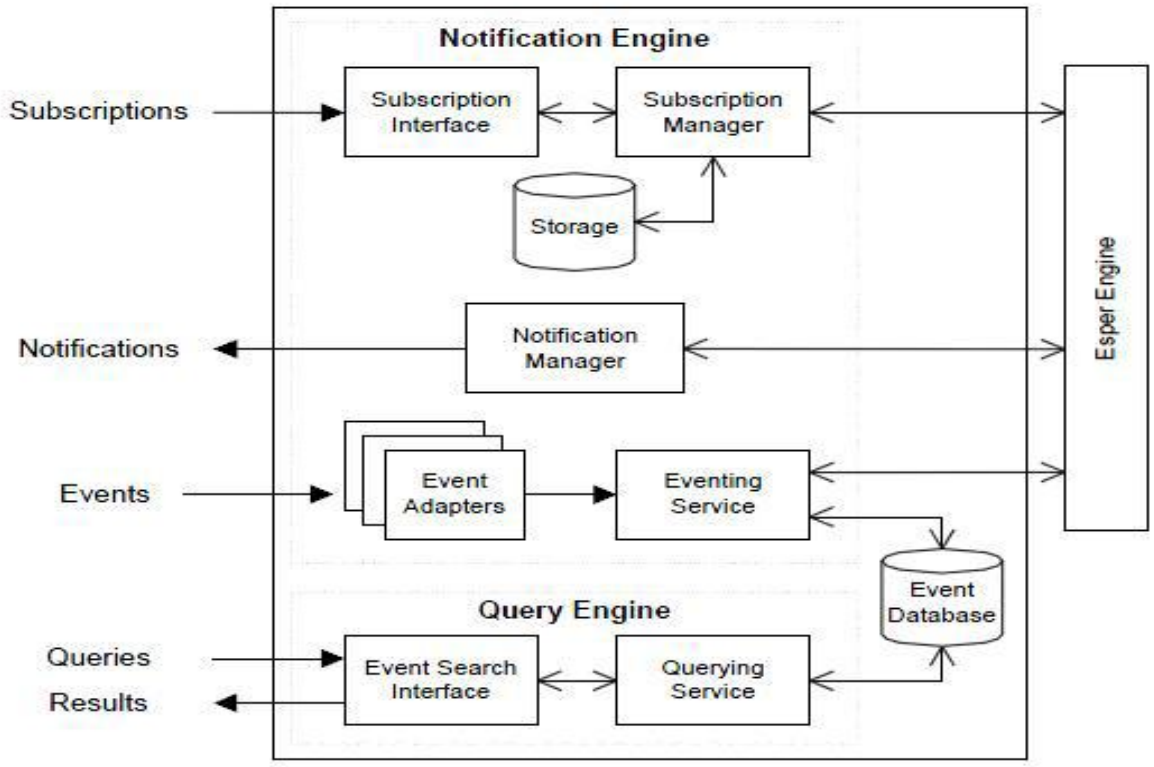


Figure 4.2 Exemple d'architecture orientée évènements [75]

- **Adaptateur** : Un adaptateur permet la gestion des diverses sources d'évènements et de conversion des évènements reçus-en des formats de données compréhensibles par l'infrastructure CEP pour faciliter son interprétation. Ensuite, il délivre l'évènement au composant concerné.

  Ils existent Plusieurs types d'adaptateurs comme les adaptateurs XML, les adaptateurs de bases de données et les adaptateurs d'alarmes.
- **Service de notification** : basé sur l'usage des communications Publish/Subscribe. Ce mécanisme permet de découpler les producteurs et les consommateurs des évènements des messages. Des produits standards existent tels que CORBA Notification Service ou Java Message Service (JMS) appliquent ce mécanisme.

- **Moteur de règles** (un moteur d'évènements) est responsable de l'exécution des règles de traitement des évènements. Une règle prend en entrée un patron d'évènements et produit un ensemble d'évènements en sortie en appliquant des opérations de traitement (transformation, corrélation, composition, etc.).
- Dépôt de métadonnées Le dépôt assure la sauvegarde des concepts et des données du système tels que les ontologies d'évènements, la configuration des adaptateurs, la configuration des services et les données des règles. Ce dépôt est utilisé pour la configuration et la maintenance [77].
- **Réseau de traitement des évènements (EPN)** : est un framework conceptuel constitué de quatre composants :
- des producteurs d'évènements,
- des consommateurs d'évènements,
- des agents de traitement des évènements (EPA) et
- des composants de connexion appelés canaux d'évènements (Eventchannel) [76].

Un EPN permet de décrire les échanges d'évènements entre producteurs et consommateurs d'évènements à travers des agents qui traitent ces évènements en appliquant des opérations comme la transformation, l'agrégation et la corrélation. Un aperçu de ces traitements d'évènements est donné dans [69] qui propose une classification des EPA en fonction des opérations qu'ils efiectuent (voir figure 2.7). Selon [78], ces agents procèdent aux trois tâches suivantes :

- détection : de patrons : pour sélectionner les évènements à traiter,
- traitement : en appliquant des opérations sur ces évènements,
- émission : pour décider comment et où envoyer les évènements dérivés.

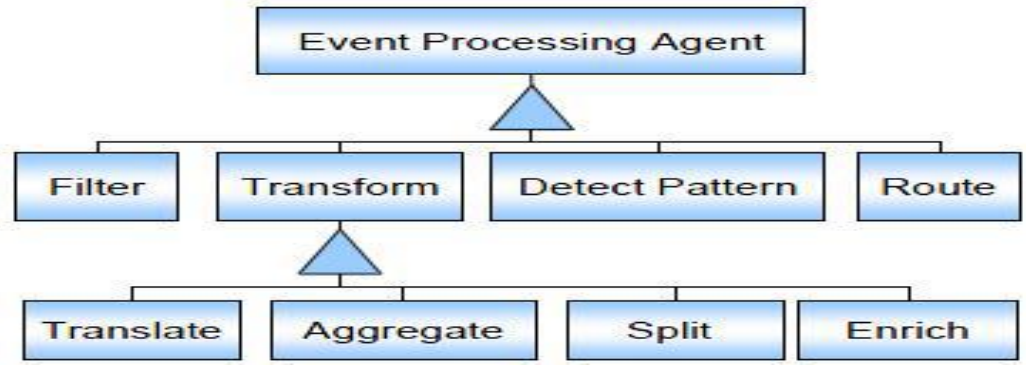


Figure 4.3 Classification des agents EPA [Lakshmanan et al. 2009]

Les réseaux EPN permettent une réutilisation des agents permettant ainsi une modification et une évolution plus facile des systèmes. De plus ces réseaux peuvent être décomposés dynamiquement en assemblant plusieurs EPN. Cette approche permet aussi de faciliter la gestion des évènements complexes en utilisant des étapes successives simples au lieu de procéder avec des patrons complexes. Une approche de stratification des réseaux EPN est proposée par [69], où il est possible d'organiser des agents, partageant des caractéristiques communes, en des strates comme le montre la figure . Cette approche a l'avantage de permettre à des agents de travailler en parallèle et d'organiser des systèmes distribués complexes optimisant ainsi le temps d'exécution et améliorant la Scalabilité de l'application [69].

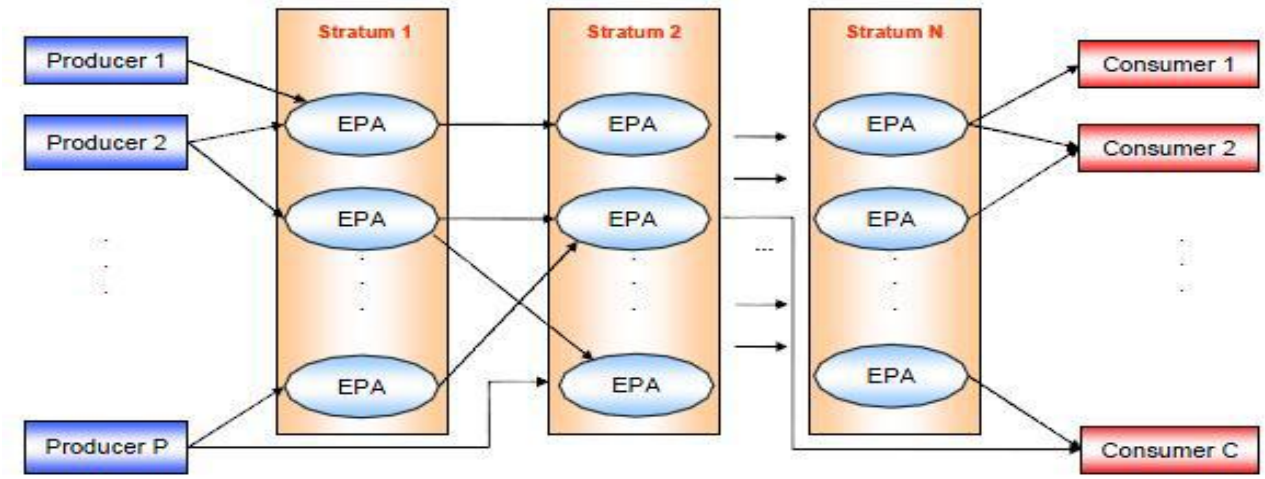


Figure 4.4 Réseau EPN stratifié [Lakshmanan et al., 2009]

Les approches orientées évènements bénéficient d'une grande importance, ceci est dû à un besoin croissant de communication asynchrone, scalables et fortement distribuée. La communication via des flux d'évènements garantit un faible couplage entre les composants distribués et une flexibilité pour supporter la dynamicité de leur topologie. Cette section présente un aperçu sur ces approches en décrivant les usages possibles des systèmes à base d'évènements, les modèles utilisés pour les évènements (concept central), les approches de traitement des évènements et les architectures ainsi que l'infrastructure d'un système à base d'évènements.

- **L'architecture orientée service** :

La notion d'architecture orientée service (SOA) est un paradigme abstrait dans le cas des SOA, ce qui n'est pas nécessairement le cas des WSOA, sans aucune référence à une implémentation technique.

L'objectif d'une architecture orientée services est de décomposer une fonctionnalité complexe d'un système en un ensemble de sous-fonctions qui sont nommées services. L'idée sous-jacente est de construire une architecture décomposée en services correspondant à différents processus métiers d'un système.[79]

Définition L'architecture **SOA** (**Service Oriented Architecture**)

Fournit un ensemble de méthodes pour le développement et l'intégration de systèmes dont les fonctionnalités sont développées sous forme de services interopérables et indépendants. Une infrastructure SOA vise à permettre l'échange d'informations entre applications. Les concepts de SOA se basent sur des concepts plus anciens, en particulier l'informatique distribuée et la programmation modulaire. [80]

Principes fondamentaux de SOA

Afin de mettre en œuvre une architecture SOA avec succès, il ne suffit pas d'avoir les capacités technologies. Il convient également d'adopter un certain nombre de principes importants au niveau de la conception, du développement et de la gestion. Dans le cadre de SOA, ces principes constituent un Framework qui va permettre aux clients et aux services de collaborer et servir d'orientation pour la conception et le développement de services. Il existe de nombreux principes qui peuvent être appliqués à une SOA : [80]

- **Couplage faible**

Le couplage est une métrique indiquant le niveau d'interaction entre deux ou plusieurs composants logiciels. Deux composants sont dits couplés s'ils échangent de l'information .On parle de couplage fort (ou serré) si les composants échangent beaucoup d'information et de couplage faible (ou relaxé) dans le cas contraire. Dans une architecture SOA, le couplage entre les applications clientes et les services doit être faible. Cela signifie qu'il y a une séparation logique qui isole le client du service afin d'éviter une dépendance physique entre les deux. Pour ce faire, le client communique avec le service par échange de messages dans un format standard. Il est ainsi possible de modifier un service, par exemple pour y ajouter des fonctionnalités, sans briser la compatibilité avec les applications clients existantes. En cas de couplage fort, la possibilité d'évolution des services serait très limitée.[79]

- **Interopérabilité**

Tout comme le couplage faible, l'interopérabilité est un critère primordial pour mettre en ouvre une architecture SOA avec succès. L'interopérabilité se définie comme la capacité que possède un produit ou système, dont les interfaces sont intégralement connues, à fonctionner avec d'autres produits ou systèmes existants ou futurs. Il convient de distinguer la compatibilité et l'interopérabilité. En effet, la compatibilité est une notion verticale qui fait qu'un produit peut fonctionner dans un environnement donné en respectant ses caractéristiques. La notion d'interopérabilité est au contraire transversale qui permet à deux produits de communiquer de part une connaissance bilatérale de leur manière de fonctionner. L'interopérabilité est rendue possible par le respect de normes et formats par tout système ou produit. Dans le cas d'une architecture SOA, l'interopérabilité permet à des applications clients de communiquer avec des services programmés dans des langages de programmation différents et s'exécutant sur des plateformes (logicielles ou matérielles) différentes. Similairement au couplage faible, la communication par échange de messages joue ici un rôle très important. En effet, l'application ne connaît pas et n'a pas besoin de connaître le fonctionnement interne d'un service pour pouvoir l'utiliser. Il est simplement nécessaire de définir un format d'échange standard entre le client et le service. Le format XML est largement accepté et pris en charge pour l'encodage des messages. En effet, la plupart des plateformes technologiques sont aptes à générer et à traiter des messages au format XML.

- **Réutilisabilité**

Le principe de réutilisabilité permet de réduire les coûts de développement en favorisant la réutilisation de parties de codes qui ont déjà été réalisées. Dans le cas de SOA, l'objectif de la séparation des tâches en services autonomes est en effet de promouvoir leur réutilisation. Ainsi, lorsque les nouveaux clients définissent leurs besoins, il est généralement possible d'utiliser certains services existants pour satisfaire une partie des besoins.

Ceci permet un gain de temps considérable et une réduction importante de la taille des applications par évitement de la duplication. Ceci facilite également la maintenance et diminue les chances de bogues dans le programme. Dans une architecture SOA, la granularité du découpage des fonctionnalités en services est également importante. Il faut que celle-ci soit suffisante pour qu'un service soit significatif et utile. Découverte La découverte est une étape importante pour permettre la réutilisation des services. Il faut en effet être en mesure de trouver un service afin de savoir qu'il existe et pouvoir en faire usage. Ainsi, même si un service fournit une fonctionnalité importante, il serait très inefficace s'il n'était pas découvrable pour être réutilisé plus tard. Une solution typique pour ce genre de problème

consiste à utiliser un annuaire de services. Un annuaire est très similaire à un catalogue ou un inventaire de services. L'annuaire contient des informations publiées concernant les services et fournit typiquement une possibilité de recherche basée sur une fonctionnalité recherchée. Il est alors nécessaire qu'un service pour SOA soit conçu pour être suffisamment descriptif pour qu'il soit facilement localisable et accessible via un mécanisme de découverte. Le développeur peut alors simplement consulter cet annuaire afin de trouver un service adéquat pour réaliser une tâche donnée et l'utiliser dans son code. [75]

- **Les types d'architectures** :

Deux possibilités, l'architecture orienté service :

- centralisée : elle s'appuie sur un service de registre de service ;
- décentralisée : elle s'appuie sur un mécanisme de découverte de service en multicast sur le réseau.

a. **Avec annuaire centralisé** (fournisseur, consommateur, répertoire)

Prenons l'exemple du fournisseur/consommateur : dans un premier temps, le fournisseur publie sa description à un service de registre (par exemple UDDI), nommé répertoire dans le schéma ci-dessus. Lorsqu'un consommateur a besoin d'un service, il recherche dans ce même répertoire les descriptions des services disponibles, il récupère la référence qui l'intéresse. Une fois la référence récupéré, il établit une communication avec le service désiré. [80]

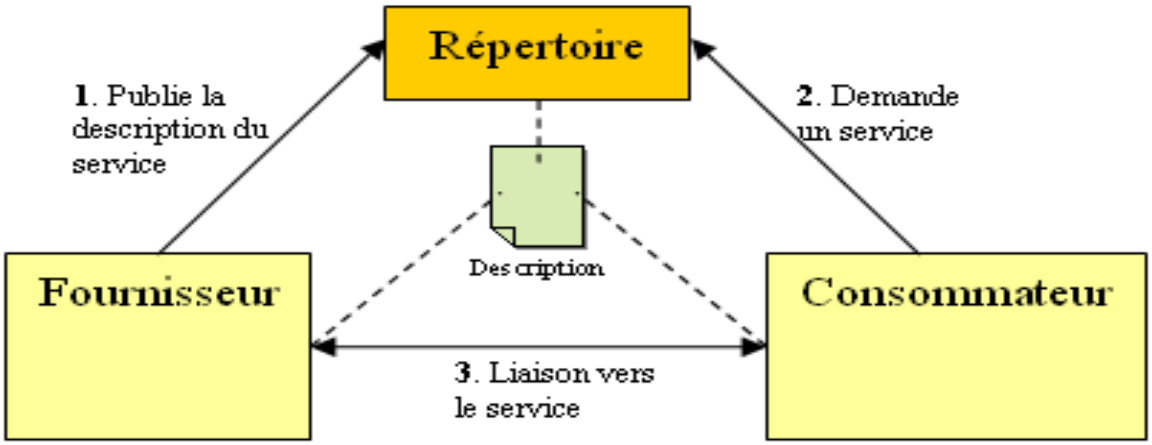


Figure 4.5 Exemple : SOA centralisée

b. Avec annuaire décentralisé :

Dans le cas d'une SOA décentralisée, la découverte de service ne se fait plus en se connectant à un service de registre. L'enregistrement dans le registre est remplacé par une annonce (message Hello) en multicast. Si un consommateur est intéressé il demande la description du

service, et établit une connexion vers le service désiré. [79] représenté dans la figure suivante :

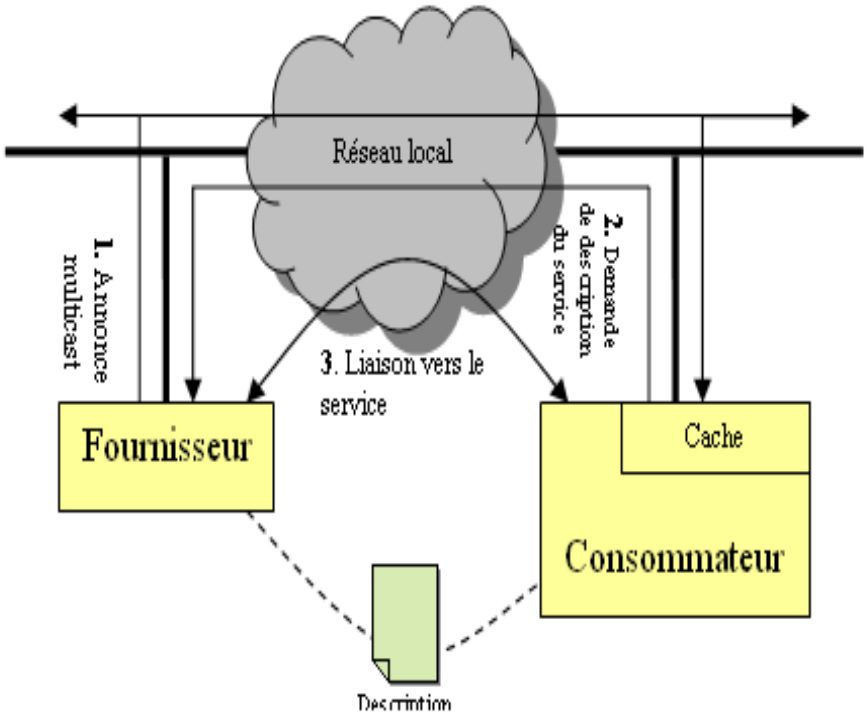


Figure 4.6 SAO décentralisé

- **L'architecture Modèle_ Vue contrôleur MVC**

Le principe de l'architecture MVC est de Sépare la couche interface utilisateur des autres parties du système (car les interfaces utilisateurs sont beaucoup plus susceptibles de changer que la base de connaissances du système).

Elle Composé de trois types de composants :

- Modèle : rassemble des données du domaine, des connaissances du système. Contient les classes dont les instances doivent être vues et manipulées
- Vue : utilisé pour présenter/afficher les données du modèle dans l'interface utilisateur
- Contrôleur : contient les fonctionnalités nécessaires pour gérer et

Architecture de MVC [59]

- **Modèle** :

C'est noyau de l'application, il permet :

- Enregistre les vues et les contrôleurs qui en dépendent
- Notifie les composants dépendants des modifications aux données

- **Vue** :

C'est l'interface (graphique) de l'application, elle offre :

a. Crée et initialise ses contrôleurs
b. Affiche les informations destinées aux utilisateurs
c. Implante les procédures de mise à jour nécessaires pour demeurer cohérente
d. Consulte les données du modèle

- **Contrôleur** :

C'est partie de l'application qui prend les décisions, elle :

a. Accepte les événements correspondant aux entrées de l'utilisateur
b. Traduit un événement (1) en demande de service adressée au modèle ou bien (2) en demande d'affichage adressée à la vue
c. Implémente les procédures indirectes de mise à jour des vues si nécessaire.

- **Les avantages d'un point de vue conception :**
  - les composants peuvent être conçus indépendamment
  - Elle assure meilleure cohésion – Couplage
  - elle offre la réutilisabilité (la vue et le contrôle peuvent être conçus à partir des composants déjà existants)
  - elle assure la flexibilité (il est facile de changer l'interface utilisateur)

- **Une architecture pervasive sécurisée: PerSE** (Laboratoire LIRIS, INSA de Lyon) [81]

La sécurité dans les environnements pervasifs est un facteur clé pour l'acceptation des technologies apparaissant dans ces environnements. L'omniprésence des dispositifs autour de l'utilisateur doit lui apporter des services utiles et pertinents en fonction de ses besoins, de manière réactive (après avoir exprimé une intention) ou de manière proactive (anticipation des besoins).Toutefois, chaque utilisateur veut pouvoir contrôler la manière avec laquelle il interagit avec son environnement, en particulier quels services ou données il est prêt à partager avec lui. En effet, il doit pouvoir décider différents niveaux d'autorisation d'accès à son environnement : par exemple, quelles données sont extrêmement confidentielles et ne devraient transiter que sur une liaison sécurisée, quelles données sont assez peu confidentielles pour pouvoir être lues par n'importe quel service ou personne alentour, quels sont les droits d'usage des services gérés sur un dispositif, Toutes sortes d'autorisations personnalisées doivent pouvoir être mises en place dans une architecture pervasive, en

fonction du contexte d'utilisation et de la confiance dans son environnement. La sécurité doit donc se trouver d'une part à l'interface entre les dispositifs pervasifs eux-mêmes, et également à l'intérieur des dispositifs pour contrôler l'accès aux données et services hébergés. De plus, l'ajout de sécurité dans le système pervasif ne peut se faire au détriment des performances, et doit rester la moins intrusive possible. La plupart des approches actuelles des environnements pervasifs n'abordent pas le problème de la sécurité au cœur de l'architecture. Cette dernière se trouve généralement à la périphérie de l'architecture globale, de manière auxiliaire, ou se limite souvent à une autorisation d'accès sommaire au dispositif pervasif. Le projet Aura se concentre sur la modélisation des tâches des utilisateurs pour créer un système proactif. 3PC permet aux applications pervasives de s'adapter à l'environnement, notamment grâce à un système de composants distribués spécifiés par contrat, de même que one.world , qui met en place un framework simplifiant la migration et la composition d'applications, ainsi que le partage de données. Certaines approches proposent cependant des architectures pervasives sécurisées, mais deviennent très limitatives et ne permettent que peu de choses dans l'environnement pervasif. Les interactions se limitent alors à des échanges de données entre dispositifs, comme dans l'architecture pawS ou Confab ..La solution Daidalos fait, elle, appel à une tierce partie de confiance pour garantir la confiance de certaines entités, ce qui pose certains problèmes dans un environnement pervasif très dynamique où une tierce partie n'est pas forcément présente.

Par rapport à des modèles de contrôle d'accès discrétionnaires, notre travail permet de prendre fortement en compte le contexte de l'utilisateur. Les modèles RBAC ou ORBAC [65], basés sur les rôles et/ou les organisations, sont eux destinés à des structures prédéfinies, dans lesquelles la dynamicité n'est pas importante, et ne prennent que peu en compte le contexte de l'utilisateur dans son environnement.

Aussi, nous pensons qu'une approche intégrée prenant en compte de multiples aspects de sécurité dès la conception d'une architecture pervasive est une solution élégante et performante.

ENVIRONNEMENT PERVASIF A BASE DE SERVICES

Il se comportant d'une manière non-intrusive et autorisant des actions réactives et proactives. Le résultat se concrétise par une plateforme nommée PerSE. Pour intégrer un environnement PerSE, chaque équipement exécute un méta-service, appelé Base, lui permettant de partager ses services locaux et son contexte local. La Base PerSE se charge des communications avec les autres Bases afin d'exécuter intelligemment des services répartis, de

manière transparente et adaptée, dans le but de répondre aux besoins des utilisateurs. Un environnement PerSE comprend ainsi plusieurs Bases autonomes capables de se découvrir et d'échanger des messages à travers différents canaux de communication (LAN, WiFi, Bluetooth, …) disponibles sur les équipements connecté de services interopérables pouvant être exécuté.[81]

- **Base PerSE :**

  assure plusieurs fonctionnalités réparties en trois niveaux : Communication, Environnement, Action (voir Figure 1).

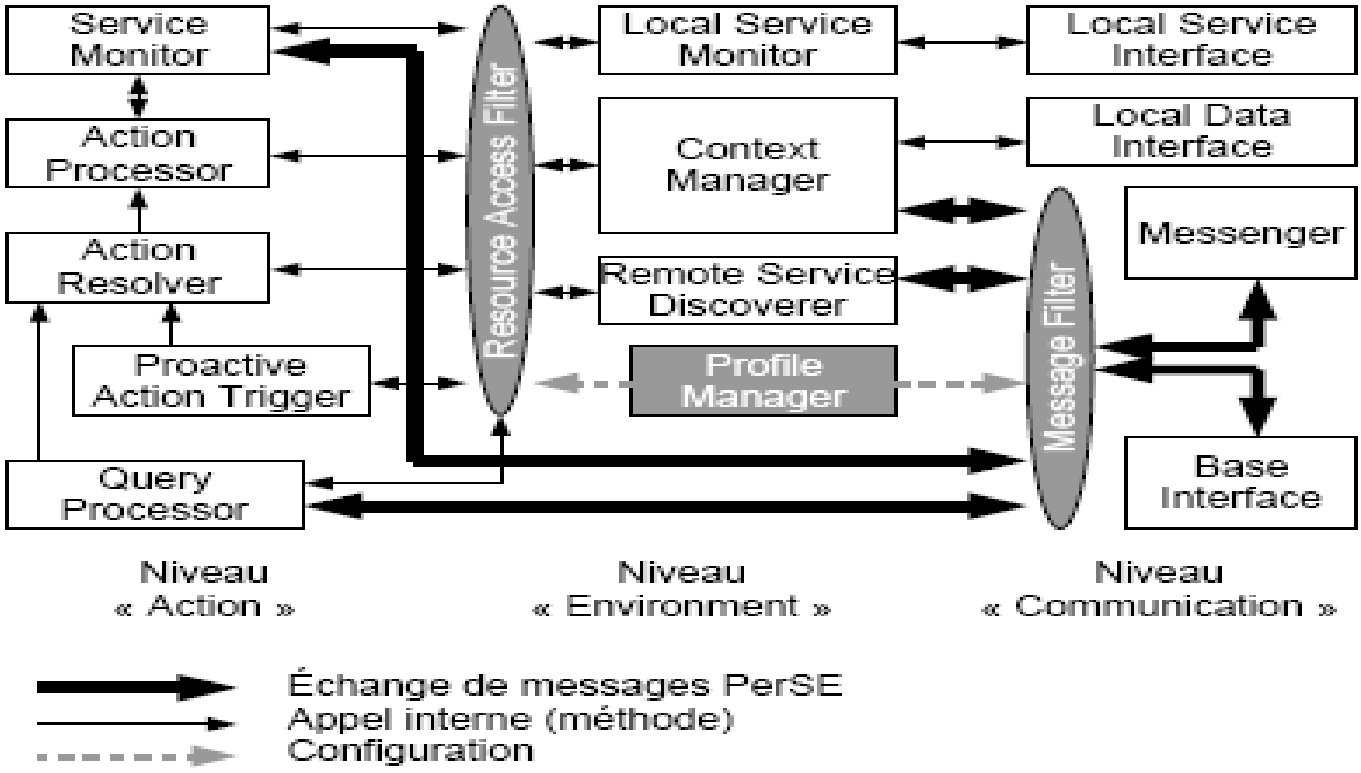


Figure 4.7 Architecture globale d'une Base PerSE

Afin de prendre en compte les besoins de sécurité, des modules spécifiques ont également été intégrés (en gris sur la figure). La communication entre modules se fait soit de manière directe (appel de méthodes), soit par échange de messages PerSE asynchrones.

A. **Niveau « Communication »**

Le niveau Communication comprend les interfaces entre la BasePerSE et l'environnement informatique (réseaux, système des fichiers, applications, …). Les modules Local Data Interface et Local Service Interface permettent d'abstraire les appels au système

d'exploitation afin d'unifier l'accès aux données locales (capteurs, fichiers) et aux services locaux (services Web, programmes, ...)

Le module Base Interface est le point d'entrée local pour les utilisateurs (ou les applications utilisatrices) voulant utiliser les fonctionnalités de la Base, notamment pour démarrer des actions partielles en utilisant PsaQL. Le module Messenger est chargé desc ommunications entre les Bases de l'environnement PerSE(découverte et échange de messages). Ces deux modules,connectés avec le monde extérieur, envoient et reçoivent (à traversle Message Filter, pour des raisons de sécurité, voir partie 3.1), des messages PerSE vers/depuis les autres modules.

B. Niveau « Environnement »

Le niveau Environnement gère les ressources locales de la Base,ainsi que ses connaissances sur l'environnement PerSE. Le LocalService Monitor permet de gérer, de manière unifiée, l'ensemble des services locaux disponibles. Le Context Manager se charge à la fois de l'accès au contexte local et de l'accès distant (sousforme de requêtes) au contexte des autres Bases, afin d'offrir aux autres modules une vision unifiée et la plus complète possible du contexte de l'environnement. Le Remote Service Discoverer soumet des requêtes aux autres Bases pour maintenir localement un annuaire des services distants disponibles dans l'environnement.Du point de vue sécurité, l'accès aux modules de ce niveau se faitpar l'intermédiaire du module Resource Access Filter, commandé par le Profile Manager, afin de filtrer les informations disponibles au niveau Action.[81]

C. Niveau « Action »

Le niveau Action se charge d'une part de recueillir les requêtes venant des autres Bases ou du module Base Interface, d'autre part d'exécuter des actions. Le Query Processor réceptionne des messages PerSE encapsulant des requêtes et y répond en recherchant les informations demandées dans le niveau environnement (liste de services, accès au contexte). Il reçoit aussi des actions partielles, déclenchant alors une nouvelle action. Le module Proactive Action Trigger, qui surveille le contexte et maintient un historique des actions réalisées, peut également générer des actions partielles de manière proactive.

Les actions partielles sont alors transmises à l'Action Resolver pour être transformées, en fonction du contexte et des services disponibles, en actions complètes pouvant être exécutées par l'Action Processor. Le Service Monitor est utilisé pour commander l'exécution de services, en local et à distance, et surveille leur statut [81].

. MODELE DE SECURITE DANS PERSE

La sécurité et la protection des ressources donnée est l'un des objectifs majeurs de l'architecture PerSE. Nous avons donc intégré dès la conception de l'architecture elle-même des modules dédiés à la sécurité. Ainsi 3 modules sont dédiés à la sécurité :

- **Le Message Filter :**

Le Message Filter agit comme un filtre sur les communications entrantes et sortantes, communications basées sur des messages asynchrones. Un message PerSE possède la structure suivante :

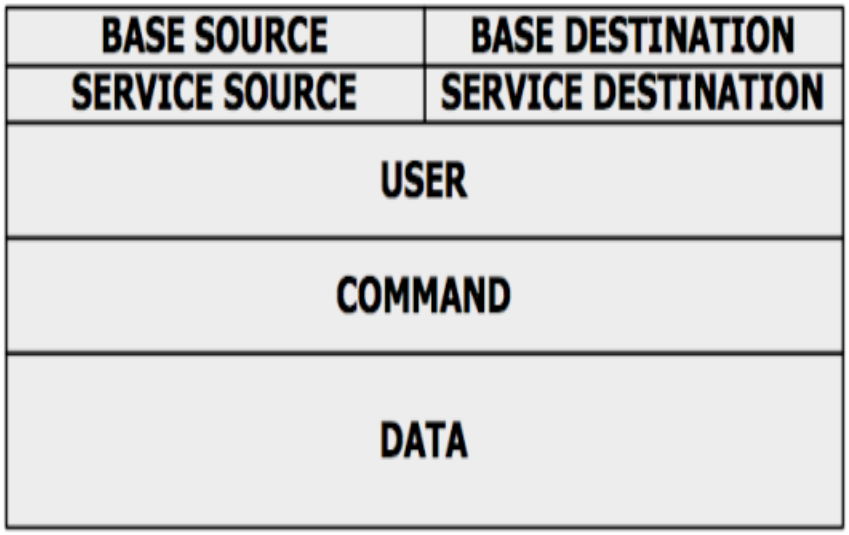


Figure 4.8 Structure d'un message PerSE

Il est composé de deux parties principales : l'entête et les données. L'entête est composé elle-même de 4 couches :

– les bases source,
– destinataire du message,
– les services sources,
– destinataire,

L'utilisateur à l'initiative du message (s'il existe), et un champ commande, qui définit le type du message : requête, réponse à une requête, découverte, etc. L'entête d'un message contient donc les informations nécessaires à son identification et à son filtrage.

Le rôle du Message Filter est d'autoriser ou non le passage de ces messages ,on appliquant les **règles de communication**. C'est donc un premier Policy Decision Point (PDP) pour l'architecture, mais aussi un Policy En forcement Point (PEP) au niveau des communications.

Une règle de communication permet d'effectuer 3 actions sur un message provenant d'une entité : accepter (allow), refuser et notifier du refus (deny), ou refuser et ne pas notifier (drop). Communication est de la forme Ces règles représentent le premier niveau de la sécurité. [81]

- **Le Resource Access Filter**

Le second niveau de sécurité se situe au niveau de l'accès aux ressources. Le Resource Access Filter permet de contrôler les interactions entre les différentes entités (base PerSE, un service, ou un utilisateur) et les ressources (données ou services). Ici encore, ce contrôle est réalisé à base des **règles**, regroupées sous forme de **profils**. À un moment donné, un profil est appliqué sur la base, et réglemente l'accès aux ressources. Le Resource Access Filter est donc un second PDP, Une **règle d'accès aux ressources** définit, pour un groupe d'entités, le droit d'effectuer une action sur des groupes de données (modification, suppression, MAJ.) ou de services (exécution, découverte, , communication entre services, etc).

Un **groupe d'entités** est défini par l'utilisateur ou l'administrateur de la base, et regroupe des entités proches du point de vue de la confiance et de la sécurité).

- **Le Profile Manager**

Le profil à utiliser est défini par le Profile Manager, qui représente véritablement le cœur de la sécurité de l'architecture PerSE. Ce module permet de déterminer quelles règles s'appliquent lors de la réception d'une requête, et ce en fonction du contexte qui entoure l'utilisateur. C'est le Policy Administration Point (PAP) de l'architecture, car il décide de la politique de sécurité à appliquer.

Figure 3 : Structure du Profile Manager). Dans chaque espace contextuel s'applique un **algorithme de détermination des règles**, transmis par l'Algorithm Manager, qui gère les différents algorithmes existants, puis exécuté par le Rule Decider. Chaque algorithme est basé sur des paramètres contextuels différents (la confiance que l'on a dans les entités présentes alentour, la localisation de l'utilisateur ou encore la température actuelle ,etc.). Ce contexte constitue **le support des règles,** c'est-à-dire l'ensemble des paramètres contextuels et leur valeur, pour lesquels la règle est valide. Nous pouvons maintenant définir une règle d'accès aux ressources. Une règle est de la forme :

Les règles font intervenir 4 paramètres : le groupe de l'entité (group_x), l'action, la ressource sur laquelle s'effectue l'action, et le support des règles, c'est à dire l'espace contextuel dans lequel la règle est valide.

De nombreuses études ([82], [83]) sur la perception de la sécurité par les utilisateurs dans un environnement pervasif ont montré que celle-ci varie énormément selon l'utilisateur, et donc

chaque personne possède ses propres critères d'évaluation de la confiance d'un lieu. Nous offrons donc à l'utilisateur la possibilité de définir lui-même de nouveaux espaces contextuels et de nouveaux algorithmes selon les paramètres qu'il souhaite utiliser dans un espace précis pour déterminer les règles et donc la politique de sécurité associée. Cet ajout est réalisé par l'Algorithme Manager, qui mettra à disposition le nouvel algorithme en fonction du besoin. Pour cela, l'utilisateur dispos d'une liste des données contextuelles à sa disposition et sur lesquelles il peut se baser pour définir ses algorithmes.

En l'absence d'algorithmes ajoutés par l'utilisateur, c'est un algorithme par défaut qui est utilisé, basé sur la localisation de l'utilisateur et son voisinage proche, lointain, et inconnu. Il est même possible de composer les algorithmes pour répondre à des besoins plus précis, par exemple à la fois sur la localisation et la température.

Les règles à appliquer, groupées sous forme de profils, sont transmises aux **Message Filter** et Resource **Access Filter** qui les utiliseront. Enfin, un historique permet de stocker les différentes situations rencontrées, ainsi que l'algorithme utilisé et les profils de règles déduites de l'algorithme, ceci afin d'éviter, pour une situation déjà, rencontrée plusieurs fois, de ré-exécuter les algorithmes et demander des données, ce qui peut parfois se révéler coûteux.

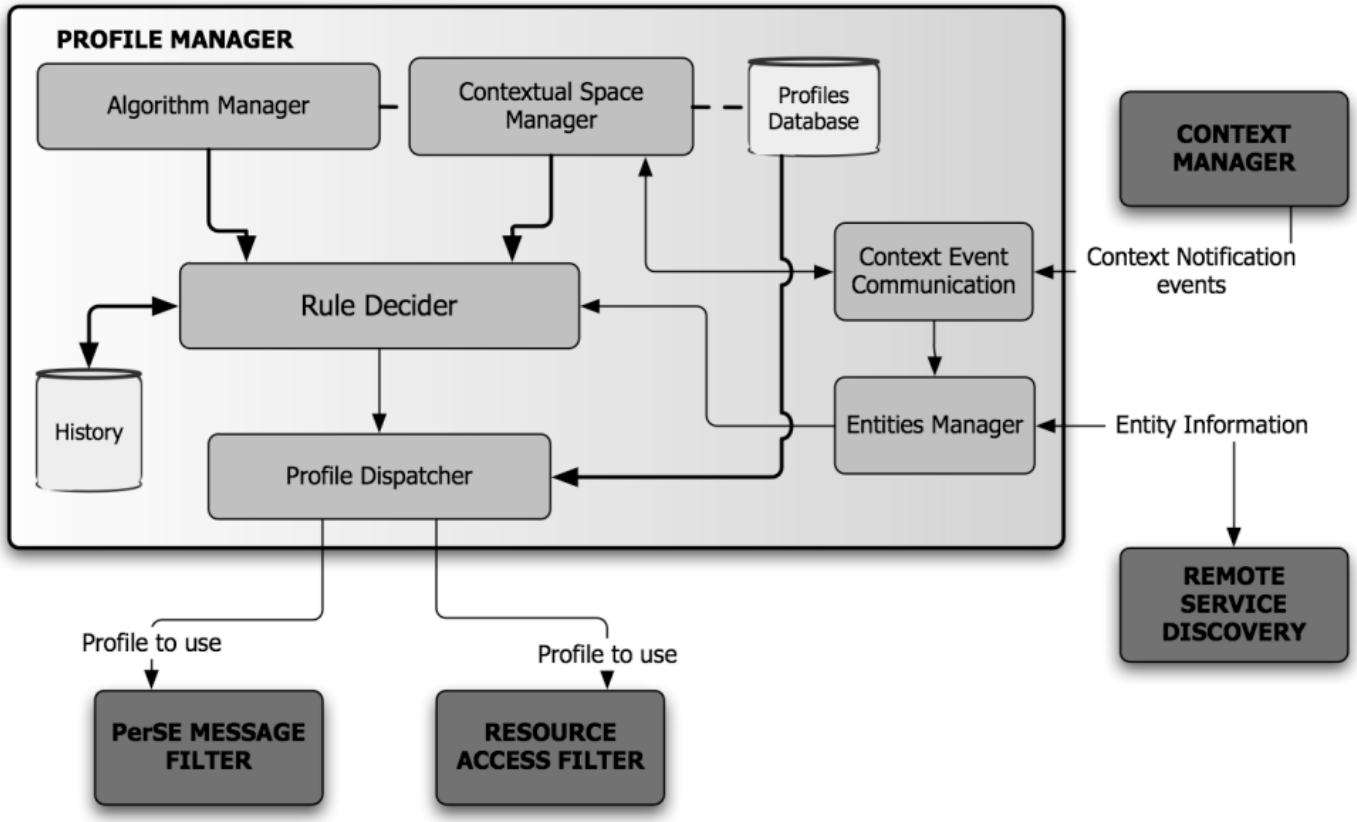


Figure 4.9 : Structure du Profile Manager

Concernant les règles utilisées dans notre modèle, l'objectif à moyen terme est de fournir pour l'utilisateur un langage de description simple, basé sur une grammaire décrite en BNF. Ce langage permettra ensuite d'interpréter et d'implémenter les règles dans un standard reconnu comme XACML, dont les règles basées sur le triplet "Sujet-Ressource-Action" correspondent à notre sémantique.

11. Conclusion

De nos jours, de nouveaux besoins en systèmes d'information sont apparus. L'utilisateur d'une application souhaite avoir l'information quelque soit le moment ou le lieu où il se trouve.

Ceci a incité les développeurs à intégrer les terminaux mobiles dans leurs applications, donnant ainsi naissance à de nouveaux systèmes d'information dits pervasifs ou ubiquitaires. Ces systèmes dits sensibles au contexte doivent avoir la possibilité de percevoir la situation de l'utilisateur dans son environnement et d'adapter par conséquent tout le comportement du système à la situation en question : ses services, ses données et son interface utilisateur, et ce, sans intervention explicite de l'utilisateur.

Dans ce genre de système, l'adaptation de l'architecture de l'application à un ensemble de paramètres (type de terminal, état de connexion, utilisateur connecté…) doit être assurée pour garantir une utilisation confortable. Tous ces paramètres forment un contexte d'utilisation particulier. Dans différents contextes, les utilisateurs accèdent aux mêmes données et aux mêmes services.

Introduction

Dans ce chapitre nous décrivons les différents enjeux de l'informatique ubiquitaire en termes de technologie, des domaines d'application et des équipements.

1. Les technologies

1.1. La biométrie

C'est-à-dire la technologie qui mesure les caractéristiques du vivant, est de plus en plus utilisée depuis une dizaine d'années, surtout dans le domaine de la sécurité. Celle-ci connaît une croissance exponentielle depuis les attentats du 11 septembre 2001, surtout que cet acte terroriste a été suivi par ceux de Madrid en 2004 et de Londres en 2005. Le terrorisme a par conséquent imposé le thème de la sécurité à tous les acteurs de la société, tant les décideurs que les simples citoyens. La biométrie s'impose donc de plus en plus aux yeux des États comme solution sécuritaire par excellence [84].

Définition : La biométrie est la mesure des caractéristiques physiques d'un individu, que ce soit ses empreintes digitales, la forme de son visage ou encore son ADN [85].

Ses caractéristiques:

- La robustesse: un attribut biométrique devrait être permanent tout au long de la vie d'une personne.

- La quantifiabilité : cet attribut doit être mesurable et quantifiable.

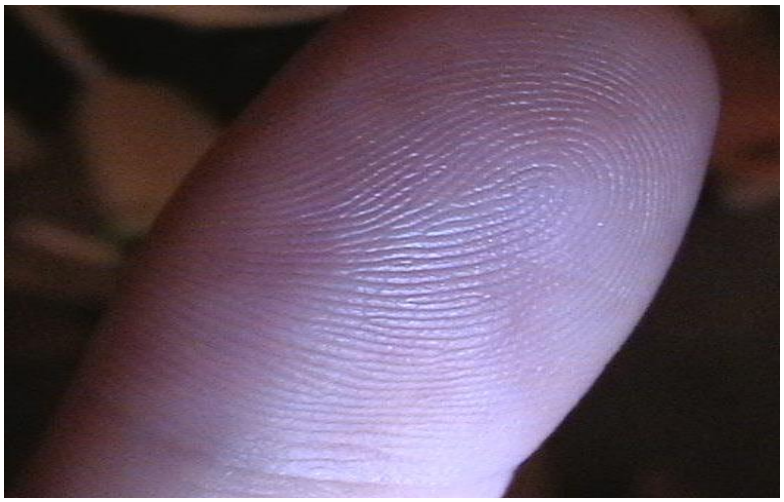
- L'acceptabilité par la population (les empreintes digitales sont souvent associées à la criminalité).

- L'universalité : quelle proportion de la population a cet attribut (certaines personnes n'ont pas d'empreintes digitales par exemple).

Les différents types d'applications :

- EMPREINTES DIGITALES (voir figure) ET DES PAUMES DES MAINS :
- GÉOMÉTRIE DE LA MAIN
- RECONNAISSANCE FACIALE
- IRIS
- RÉTINE

- VEINES
- PAR LA VOIX
- PAR LA SIGNATURE
- L'ADN



1.2. **Réseaux sans fil :**

. Un réseau sans fil (en anglais **Wireless network**) est un réseau dans lequel au moins deux terminaux peuvent communiquer sans liaison filaire. Les réseaux sans fil sont basés sur une liaison utilisant des ondes radio_électriques en lieu et place des câbles habituels.

L'installation de tels réseaux ne demande pas de lourds aménagements des infrastructures existantes comme c'est le cas avec les réseaux filaires (creusement de tranchées pour acheminer les câbles, équipements des bâtiments en câblage, goulottes et connecteurs), ce qui a valu un développement rapide de ce type de technologies.

Le périmètre géographique définissant l'étendue d'un réseau sans fil permet de distinguer plusieurs catégories de réseaux sans fil : les réseaux sans fil personnels, locaux, métropolitains et étendus. [85]

Les différents types : [86]

- Le réseau personnel sans fil (WPAN)

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, ou un assistant personnel, …) à un ordinateur sans liaison filaire. Plusieurs

technologies sont utilisées pour les WPAN dont la principale est la technologie Bluetooth, lancée par Ericsson en 1994 et proposant un débit théorique de **1 Mbps** pour une portée maximale d'une trentaine de mètres. Bluetooth, connue aussi sous le nom **IEEE 802.15.1,** possède l'avantage d'être très peu gourmande en énergie, ce qui la rend particulièrement adaptée à une utilisation au sein de petits périphériques. La nouvelle norme **blutooth SIG** dite "**bluetooth haut débit**" qui devrait être finalisée à la fin du premier semestre 2007
vise désormais l'échange de fichiers multimédia grâce à un nouveau débit, porté à 100 Mb/s dans un rayon inférieur à 10 mètres et jusqu'à 480 Mb/s pour des connexions inférieures à un mètre.[86]

- **Le réseau local sans fil (WLAN)**

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:

- Le wifi (ou IEEE 802.11), offre des débits allant jusqu'à 54Mbps sur une distance de plusieurs centaines de mètres5
- HiperLAN2 (**HIgh Performance Radio LAN 2.0**), norme européenne élaborée par l'ETSI (European Telecommunications Standards Institute). HiperLAN2 permet d'obtenir un débit théorique de **54 Mbps** sur une zone d'une centaine de mètres.

- Le réseau métropolitain sans fil (WMAN)

Le WMAN pour Wireless Métropolitain Area Network est connu sous le nom de Boucle Locale Radio (BLR). Les WMAN sont basés sur la norme IEEE 802.16. La boucle locale radio offre un débit utile de 1 à 10 Mbit/s pour une portée de 4 à 10 kilomètres, ce qui destine principalement cette technologie 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 réseau étendu sans fil (WWAN)

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 pour les réseaux cellulaires sans fil.

Selon l'infrastructure :

Les environnements mobiles sont des systèmes composés de sites mobiles et qui permettent à leurs utilisateurs d'accéder à l'information indépendamment de leurs positions géographiques. Les réseaux mobiles ou sans fil, peuvent être classés en deux classes [83] :

- les réseaux avec infrastructure
- les réseaux sans infrastructure.

1.3. Identification par radiofréquence

Technologie d'identification utilisant la communication par fréquence radio, les marqueurs RFID correspondent à un couple puce/antenne presque indécelable à l'il nu, apposé sur un produit. Ou un ensemble de produits. Permettant l'identification à distance, grâce à un lecteur qui capte les informations contenues dans la puce. Un numéro de série, une description sommaire ou un numéro de lot par exemple.

Les marqueurs (voir figure) peuvent être de deux types [82] : actif ou passif.

- Les **marqueurs actifs** contiennent une batterie interne qui permet à la puce d'être alimentée et de diffuser un signal à destination du lecteur.
- Les **marqueurs passifs** n'ont pas de batterie, c'est le signal électromagnétique du lecteur qui active le *tag* et lui permet de fonctionner. En y induisant un courant.

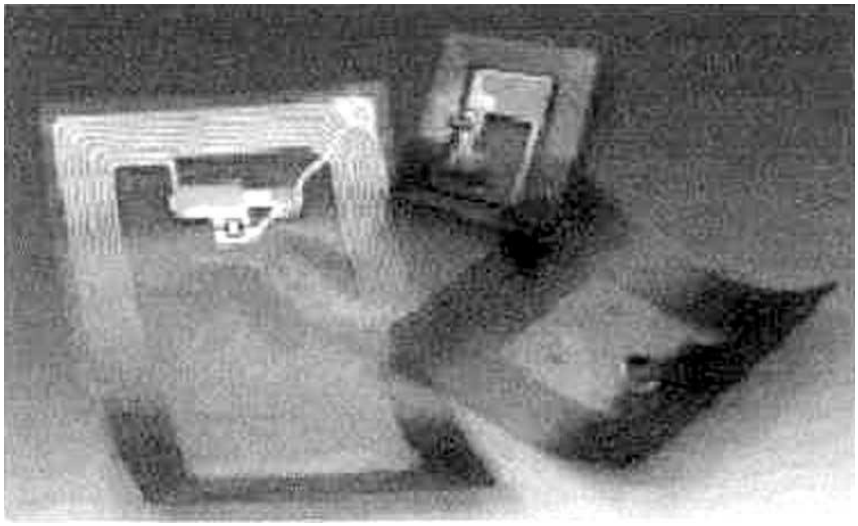


Figure5.1 image d'un marqueur RIFD

1.4. Interface homme machine :

Ensemble des dispositifs matériels et logiciels permettant à un utilisateur d'interagir avec système interactif représente dans la figure suivante

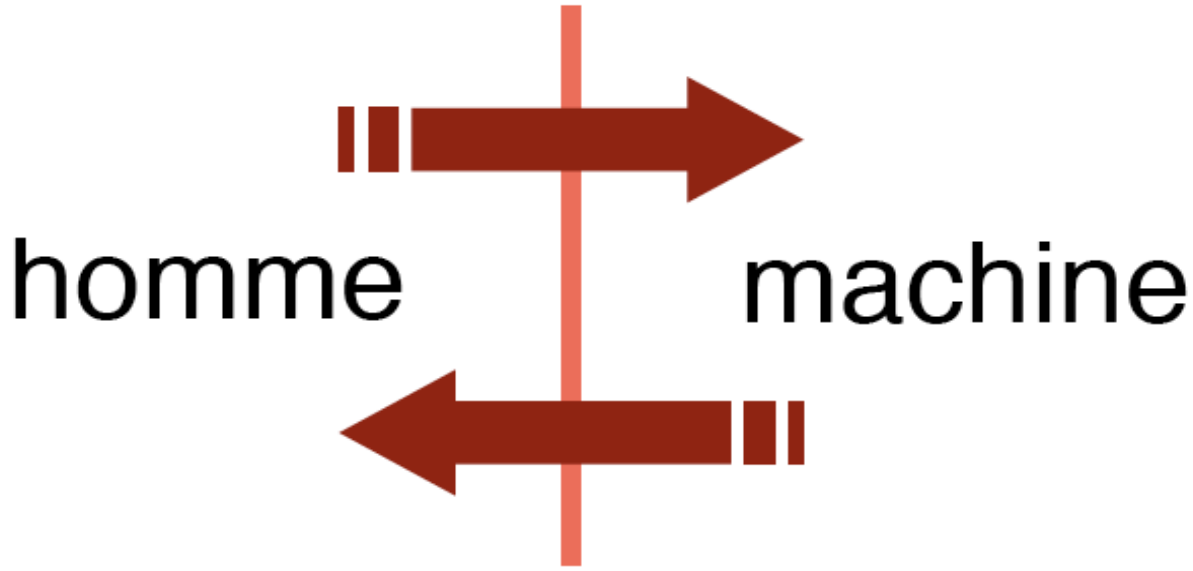


Figure 5.2 représente l'interaction homme machine

Les interfaces intelligences d'interaction homme machine sont basées sur plusieurs types d'interactions :

- . **Sensorielle** : basée sur des technologies de capture et de restitution du mouvement et des interfaces hépatiques, ainsi que des systèmes d'analyse et de synthèse sonore et olfactive ;
- . **Linguistique** (parole, texte, écriture) : basée sur des technologies de reconnaissance et de synthèse de la parole et de l'écriture entre autres [81] ;

1.5. Bluetooth

La norme Bluetooth (pris en charge par IEEE 802.15.1) est une technologie de moyen débit, elle permet d'atteindre un débit maximal théorique de 1Mbps (environ 720Kbps effectif) à basse consommation énergétique. Bluetooth utilise la bande de fréquence 2.4GHz avec une couverture entre 10 et 30 mètres.

Cette technologie permet de créer un réseau de 8 appareils en communication simultanée. La petite taille des composants Bluetooth lui permet d'être inséré dans des équipements tels que les claviers et les souris sans fil, les kits main libre ou écouteur et le transfert de données entre un pc et les PDA (Personal digital assistant) ou téléphones mobiles… [88].

1.6. Liaisons infrarouges :

La technologie infrarouge ou IrDA est également utilisée dans les applications ubiquitaire.

Cette technologie est cependant beaucoup plus sensible que Bluetooth aux perturbations lumineuses et nécessite une vision directe entre les éléments souhaitant communiquer, ce qui la limite bien souvent à un usage de type télécommande [88]

1.7. GPRS :

Le GPRS (General Packet Radio Services) est une technologie de radiocommunication par commutation de paquets pour les réseaux de GSM. Les connexions des services de GPRS sont toujours ouvertes afin d'offrir aux utilisateurs des terminaux mobiles une disponibilité de réseau identique à celle qu'ils pourraient atteindre par des réseaux d'entreprise. Le GPRS offre une connectivité d'IP de bout en bout, du terminal GPRS jusqu'à n'importe quel réseau IP.

Les terminaux peuvent être intégrés efficacement aux réseaux Internet. La vitesse "utile" sera d'environ 40 Kb/s (vitesse maximum : 171 Kb/s), l'un ou l'autre est quatre fois supérieure à celle du GSM

1.8. Les électroniques embarquées :

Du tout analogique, l'électronique embarquée est entrée progressivement dans l'ère du numérique. La microélectronique est caractérisée par une miniaturisation de plus en plus poussée avec l.intégration de plus en plus élevée de circuits « actifs » sur une même puce de silicium, une réduction du coût d.une fonction élémentaire et du transistor, une augmentation du rendement de fabrication et une amélioration de la fiabilité, par conséquent la diminution du nombre de pannes.

L'utilisation de nouveaux substrats pour les puces (arséniure de gallium), le développement de l'électronique moléculaire et la nanotechnologie bouleverseront la microélectronique dans les années à venir.

DRM (**Device Relationship Management**) permet de surveiller, maintenir et diriger en temps réel via Internet des appareils intelligents ayant des capacités de traitement de l'information déployés à distance. [89] ;

1.9. **Réseaux de capteur sans fil :**

Les réseaux de capteurs sans fil ou WSN (Wireless Sensor Networks) constituent une catégorie particulière de réseaux ad hoc. Ces derniers sont conçus pour répondre à des problématiques de communications où l'homme est souvent un acteur principal (accès à un réseau global comme Internet, téléphonie, télécommande. . .). Les WSN offrent des moyens de communication très souvent spontanés entre objets autonomes, généralement sans aucune intervention humaine. Il existe des différences considérables entre les réseaux de capteurs et

les réseaux ad hoc, donc dans la plus part des cas on ne pourra pas utiliser les mêmes protocoles **[2]**.

Un réseau sans fil de capteurs est une collection de nœuds. Chaque nœud se compose d'une unité de traitement (un ou plusieurs microcontrôleurs, CPU), peut contenir plusieurs types de mémoire (RAM, disque durs et mémoires Flash), doter d'un émetteur/récepteur et une source d'énergie (par exemple, des batteries et des piles solaires). Les nœuds de ces réseaux consistent en un grand nombre de capteurs capables de récolter et de transmettre des données environnementales d'une manière autonome, dispersés aléatoirement à travers une zone géographique (champ de captage) et mettant en œuvre un routage multi saut jusqu'au nœud considéré comme un « point de collecte ». Les réseaux sans fil de capteurs se composent de nœuds de capteurs qui doivent coopérer à l'exécution d'une fonction spécifique.

En particulier, avec la capacité des nœuds de sentir, traiter et communiquer les données, elles sont bien convenues pour exécuter la détection d'événement, qui est clairement une application en avant des réseaux sans fil de capteurs

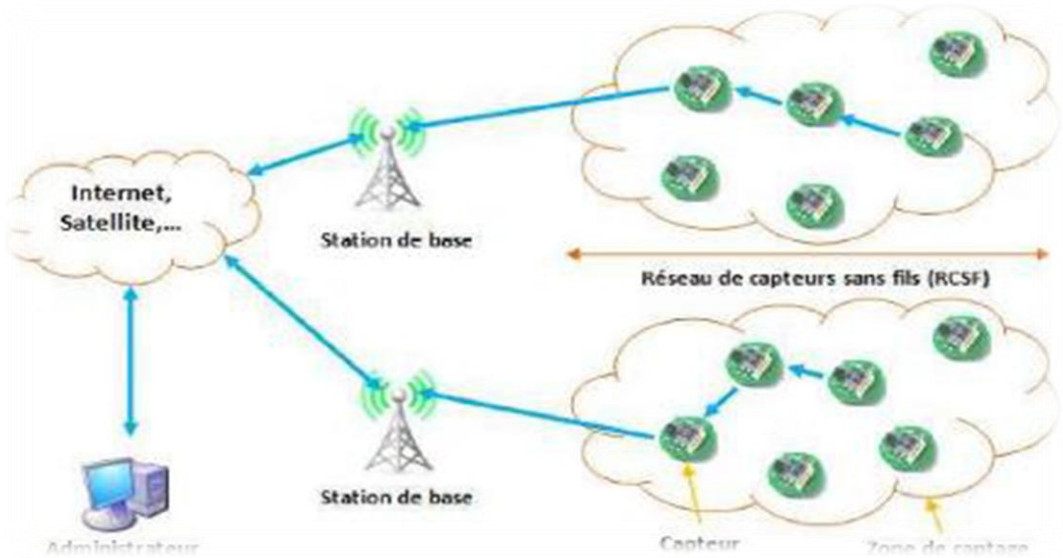


Figure5.3 illustre un réseau de capteur sans fil.

Un ensemble de caractéristiques sont importantes pour l'accomplissement des tâches assignées aux applications. Les plus importants sont :

Le Type de service : on s'attend à ce que le RCSF (réseaux de capteurs sans fil) offre à l'utilisateur, des informations significatives sur l'objet d'intérêt.

La Qualité de service (QoS) : C'est une métrique de la qualité de service qui va être offerte par un RCSF à ses utilisateurs/applications. Le niveau de **QoS** est défini par un ensemble

d'attributs comme le temps d'attente, la largeur de bande, et la perte de paquets qu'on relie directement avec le type de service du réseau. Dans **QoS** pour les **RCSFs**, la quantité et la qualité d'information extraites à partir des puits deviennent appropriées.

Tolérance aux fautes : il est important que le RCSF soit capable de traiter l'échec des nœuds capteurs. Une manière reconnue de satisfaire cette contrainte est de réaliser un déploiement redondant des nœuds capteurs.

La durée de vie : c'est la durée pendant laquelle le réseau reste opérationnel. On s'attend à ce que le RCSF puisse fonctionner au moins pendant le temps requis pour accomplir la tâche donnée. Néanmoins, la définition de la durée de vie dépend de l'application du RCSF et elle est en relation directe avec le fonctionnement efficace du réseau.

2. Les domaines des applications ubiquitaire

Plusieurs domaines s'orientent de plus en plus vers l'adoption des applications ubiquitaires pour gagner la délité de leurs clients en leur proposant des services mobiles et intelligents qui nécessitent de moindres efforts de la part de l'utilisateur. Les applications sociétales sont parmi les applications qui peuvent bénéficier le plus des technologies ubiquitaires. Nous nommons une application sociétale toute application proposant des services ayant trait à la vie quotidienne. Ces applications proposent des services personnalisés concernant différents domaines comme la santé, les loisirs, les voyages, les banques et le transport et la gestion des catastrophes naturelles.

.

2.1. Le domaine des transports en commun :

Comme de nombreux domaines, le domaine du transport présente un besoin croissant d'une gestion des évènements pour faire face à des situations critiques ou des incidents, et permettre de fournir une information en temps réel ou presque réel à l'usager.

Exemple : une application qui permet au voyageur de communiquer avec l'application et de gérer son profil et de recevoir des notifications du SIT qui peuvent l'intéresser. Par son intermédiaire, le voyageur peut consulter les informations concernant son profil et les modifier. De plus, il a la possibilité de définir son itinéraire courant parmi une liste d'itinéraires habituels, ce qui lui permet d'être informé d'éventuelles perturbations en fonction du moyen de transport emprunté lors de son déplacement dans le réseau.[89]

2.2. Les applications ubiquitaires dans le domaine médical :

- **.La télémédecine au secours des urgences**

L'urgence est un combat contre le temps. Depuis l'an 2003, l'équipe du SAMU (Service Ambulatoire d'urgences) d'Avignon utilise le service "Mobile Urgence Médicale", une solution pilote mise au point par **France Télecom** en partenariat avec la société de distribution de matériel médical **GardioGap (1).** Les services du SAMU sont ainsi informés en temps réel de l'état de santé d'un patient embarqué dans une ambulance et de ses conditions de transport pendant son transfert vers les urgences grâce à l'utilisation successive des technologies **bluetooth,** GPRS, ADSL et VPN.

L'ambulance est équipée d'un module de faible encombrement pour le recueil des paramètres vitaux (comme l'électrocardiogramme, la pression artérielle, etc.) et d'un **tablet PC** doté d'une liaison GPRS. Les deux sont reliés par une liaison Bluetooth. Au centre de régulation du SAMU, un ordinateur équipé d'une connexion ADSL permet de recevoir les données transmises par l'ordinateur de bord de l'ambulance.

Pour préserver la confidentialité et la sécurité des informations échangées à distance, la solution fonctionne sous VPN, un réseau privé virtuel, qui permet de crypter les informations et d'authentifier les intervenants. Ceux-ci ont également la possibilité de communiquer entre eux en temps réel. Ainsi, le médecin qui se trouve dans l'ambulance peut bénéficier, au besoin, de l'assistance de ses confrères présents à l'hôpital.

Les principales applications du "Mobile Urgence Médicale" touchent la cardiologie (infarctus du myocarde et syndrome coronarien aigu), la traumatologie (le recueil d'information lors d'un AVP), et le suivi à distance des patients en état grave lors des transports inter hospitaliers.[91]

- **Le dossier médical informatisé dans la poche**

Le dossier médical informatisé (DMI) est une source d'information qui doit être accessible au médecin en permanence, en tout temps et en tout lieu. Pour pouvoir conserver une image parfaitement identique du DMI à la fois sur l'ordinateur du cabinet et sur l'assistant numérique emporté en consultations, un logiciel de synchronisation fiable est requis. C'est ce qui a été réalisé en 2002 par la firme belge **SA Euremis** à travers le produit M**ediPocket**. La plupart des logiciels agréés pour le DMI sont supportés (Epicure, **HealthOne**, Le Généraliste, **MediWin Gold**, **OmniPro, Pricare,** Socrate, etc). Une partie du logiciel **MediPocket** est installée sur l'ordinateur de bureau, l'autre sur le PDA. Une première synchronisation peut avoir lieu juste après l'installation. Par la suite, une synchronisation automatique se fait entre les données du PDA et de l'ordinateur de bureau chaque fois que l'assistant numérique est

placé sur la station d'accueil. Elle se limite toutefois à ajouter des données dans les dossiers plutôt qu'à transférer ou remplacer toutes les données existantes.

D'autre part, l'application est conviviale et facile à manipuler. Les concepteurs ont en effet tenu compte des limitations de la lecture à l'écran et de la saisie d'informations, les menus de sélection et les codes ont été essentiellement privilégiés. La saisie d'un texte sur le minuscule clavier, opération loin d'être pratique, est donc réduite au strict minimum. [93]

- **L'assistance médicale pour diabétiques**

L'opérateur français SFR en partenariat avec l'association française des diabétiques a lancé le service "T+diabète». Le principe du service consiste à embarquer sur l'ordinateur de poche ou le PDA du patient une application permettant de lire, sauvegarder et analyser son taux de glycémies. Un taux mesuré par le malade lui-même via un lecteur connecté au mobile par liaison sans fil Bluetooth. Le portable est appréhendé comme un "lien permanent" entre le patient et le médecin, car les relevés sont transmis automatiquement à l'ordinateur du praticien par les réseaux de téléphonie mobile, en l'occurrence le réseau 3G de SFR. Le médecin a ainsi une connaissance détaillée du dossier de son patient grâce à l'historique des relevés. Il peut observer l'horodatage des mesures, la glycémie à jeun ou non, le nombre de doses d'insuline prises, etc. Il a également la possibilité d'envoyer un SMS pour conseiller son patient s'il constate, par exemple, que le taux de glycémie est mal suivi. De son côté, le patient peut visualiser, à tout instant, les histogrammes et les courbes d'évolution de sa glycémie. En indiquant à l'application embarquée sur le mobile l'activité physique prévue pour la journée ainsi que ce qu'il compte manger, il obtiendra par ailleurs la dose d'insuline optimale à s'injecter. [93]

- **Le bracelet GPS pour les malades d'Alzheimer**

La téléphonie mobile et la géo localisation ont été rassemblés au sein d'une solution de géo localisation à vocation médicale. Il s'agit d'un bracelet GPS développé conjointement par l'opérateur Orange et la société canadienne Médical mobile (2), aidant à localiser les patients atteints par la maladie d'Alzheimer qui, désorientés, sont incapables de retrouver leur chemin dés qu'ils s'éloignent de leurs repères familiers. Avant de s'équiper du bracelet, le responsable du malade doit définir un périmètre diurne et un périmètre nocturne à la rue près. Si le patient le franchit, le téléopérateur de médical Mobile est averti automatiquement de sa position grâce au récepteur GPS et aux antennes relais. Il alerte par un coup de fil l'entourage du patient, dont le ou les numéros de téléphone ont été enregistrés, précisant la rue où se trouve le

malade. Le responsable du patient peut même lui parler directement puisque le bracelet est muni d'une carte SIM. [93]

2.3. Les applications ubiquitaire dans les catastrophes naturelles :

L 'informatique pervasif s'oriente de plus en plus su r l'aspect de la gestion catastrophes naturelles afin d'assurer une intervention rapide et efficace.

Exemple :

- création d'une base de donnée lors d'un séisme à base des communications satellitaires pour informer les décideurs de la situation réelle dans un temps rapide, pour faciliter l'intervention et la prise en charge des victimes.

2.4. Domaine militaire

La transmission sécurisée est un des aspects principaux de toutes opérations militaires réussies. En outre, beaucoup d'opérations de la défense ont lieu dans des endroits où l'infrastructure de transmission n'est pas disponible. L'utilisation des réseaux sans fil ad hoc, de capteur et les technologies GPS dans de telles situations devient très utile. Les différentes unités (armée terrestre, marine et l'armée de l'Air) impliquées dans des opérations militaires doivent également garder la transmission entre eux. Les avions de l'armée de l'air volant dans un groupe peuvent établir un réseau sans fil ad hoc pour communiquer entre eux et échanger des images et des données. Les groupes d'armée en mouvement peuvent également utiliser les capteurs pour la localisation des positions des troupes ennemies [90].

2.5. Le Web ubiquitaire

Le Web ubiquitaire est vu alors, comme synthèse de l'Internet et l'informatique ubiquitaire qui étend le Web existant l'importance de la coordination de dispositifs mobiles, la sensibilité au contexte, l'adaptation à l'hétérogénéité des sources d'information. Le Web ubiquitaire cherche donc à élargir les capacités des navigateurs Web afin de permettre de nouveaux types d'applications Web, notamment celles nécessitant une coordination avec d'autres terminaux et une adaptation dynamique au contexte de l'utilisateur. Ces applications seront capables d'exploiter les services en réseau pour élargir les capacités des terminaux. Les utilisateurs pourront alors se concentrer sur ce qu'ils font et non sur les terminaux eux-mêmes. [91]

- **Caractéristiques de l'environnement du Web ubiquitaire**

Le *Web ubiquitaire* se fonde en premier lieu sur une connaissance de la situation, appelée communément contexte. L'étude de l'adaptation au contexte nous amène à étudier les travaux de recherche qui se sont intéressés au contexte à fin d'en dégager les concepts essentiels. Cet environnement se divise en trois catégories [91] :

- l'environnement matériel qui représente les ressources du matériel comme la bande passante de la connexion réseau, la disponibilité de la mémoire et la charge du processeur...etc.) ;
- l'environnement de l'utilisateur qui représente la situation de l'utilisateur comme son emplacement, les personnes qui l'entourent, sa situation sociale ou son état émotionnel ;
- l'environnement physique ; telle que, la localisation géographique, la lumière ou le bruit. [91]

2.6. Secteur de la grande distribution

L'informatique pervasive peut être utilisée dans L'espace de vente pour améliorer l'interaction avec les clients et les entrepôts de stockage afin de contrôler la chaîne d'approvisionnement, pour offrir :

- identification du consommateur, du contenu de son caddie et la connaissance de la dimension temporelle permettent de communiquer plus fréquemment avec le client et de fournir des services personnalisés, par exemple :
- identification de la position du client afin de proposer des produits alternatifs ou promotionnels qui se trouvent près de lui ;
- identification du contenu du caddie pour proposer des idées de recettes et de menus.
- Pour la gestion des retours et des garanties, l'historique du produit permet au personnel de savoir :

❑ quand et où il a été vendu.

❑ le prix de vente exact.

Afin de réduire le nombre de retours invalides et permettre un service au client plus rapide, moins frustrant pour le client et plus efficace.

En ajoutant des produits, identifiés de façon unique par des marqueurs RFID, dans un chariot équipé de lecteurs RFID, le consommateur a accès, par un écran de visualisation associé au chariot, au détail du contenu de son chariot et au montant des achats ainsi qu'à d'autres

informations. Lorsque le consommateur a terminé ses courses, le chariot informe directement le caissier de son contenu, un ticket de paiement est émis et le niveau d'inventaire est mis à jour automatiquement.

Exemple :

Lors d'un vol, le personnel du magasin est alerté lorsque le produit quitte le magasin. La nouvelle solution identifie les comportements suspicieux au niveau des étagères : enlever simultanément cinq CD identiques par exemple. Les produits à haute valeur peuvent déclencher une alerte de sécurité avec l'ajustement automatique des caméras de surveillance vers l'endroit où l'action a lieu. [1]

3. Les équipements

Le développent d'une technologie sera a suivie automatiquement d'un développement des équipements nous illustrons quelle que équipements de l'informatique ubiquitaire.

3.1. Le PDA: Personal Digital Assistant

- (littéralement assistant numérique personnel), aussi appelé organiseur, est un ordinateur de poche sous forme de boîtier compact de petite taille, composé d'un processeur, de mémoire vive, d'un écran tactile (manipulable à l'aide d'un stylet) et de fonctionnalités réseaux. Le PDA est utilisé principalement pour ses fonctions d'agenda, de répertoire téléphonique, de bloc-notes, de dictaphone, de lecteur MP3 et de lecteur d'images et de vidéos. [93]

3.2. Le Smartphone:

Est un téléphone portable couplé au PDA. Il offre en plus, des fonctions comme la navigation web, la consultation de courrier, la connectivité à un client de messagerie instantanée, la visualisation de fichiers PDF ou office et la navigation à l'aide du système de géolocalisation GPS. [93]

3.3.Le Tablet PC:

Un ultraportable équipé de stylet, permettant d'écrire ou de dessiner manuellement à l'écran, comme sur un bloc-notes de format A4. Il est muni d'un système de reconnaissance de l'écriture naturelle et parfois de reconnaissance vocale. [93]

Il est disponible sous deux formes :

- Convertible, c'est à dire sous forme d'un ordinateur portable traditionnel avec clavier contenant la majeure partie de l'électronique, mais dont

l'écran peut tourner et se rabattre sur le clavier, de telle sorte que seul l'écran soit visible et l'on peut interagir avec un stylet.

- Slate (ou ardoise), c'est-à-dire sous la forme d'un écran qui contient toute L'électronique, avec un stylet pour interagir [93].

3.4.Un capteur :

Est un organe de prélèvement d'informations qui élabore à partir d'une grandeur physique *(Information entrante)* une autre grandeur physique de nature différente *(Information sortante : très souvent électrique).* Cette grandeur, représentative de la grandeur prélevée, est utilisable à des fins de mesure ou de commande. Il compose des quatre unités représenté dans la figure suivante. [94]

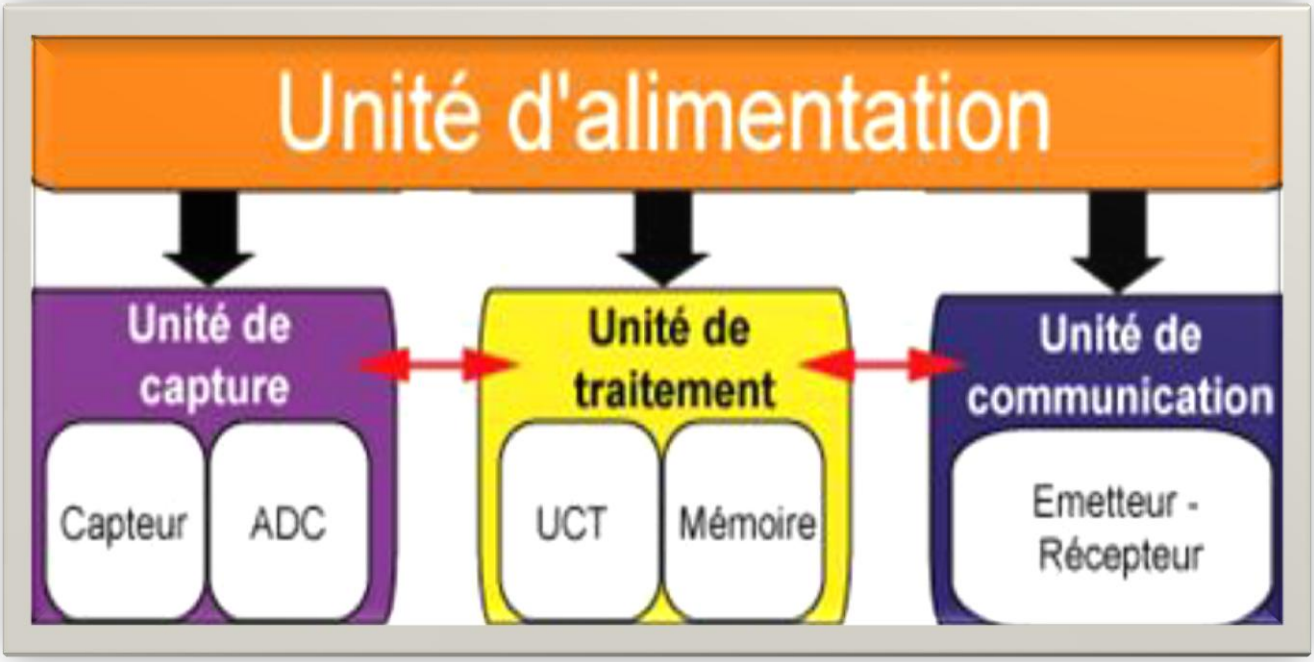


Figure 5.4 l'architecture d'un capteur

- **L'unité de traitement** : c'est l'unité principale du capteur, généralement un processeur couplé à une mémoire vive. Son rôle est de contrôler le bon fonctionnement des autres unités. Sur certains capteurs elle peut embarquer un système d'exploitation pour faire fonctionner le capteur. Elle peut aussi être couplée à une unité de stockage.
- **L'unité de capture :** elle permet la capture des données, c'est à dire la mesure des grandeurs physiques ou analogiques et leur conversion en données numériques. Elle est composée du capteur lui-même et de l'ADC (Analog to DigitalConverter) . Le capteur est chargé de récupérer les signaux analogiques qu'il transmet à l'ADC qui a

pour rôle de transformer et de communiquer les données analogiques en données numériques compréhensibles pour l'unité de traitement.

- **L'unité de communication** : elle a pour fonction de transmettre et recevoir l'information. Elle est équipée d'un couple émetteur/récepteur. Elle permet la communication au sein du réseau, dans le cas qui nous intéresse par radiofréquence (ondes radios). Il existe cependant d'autres possibilités de transmission (optique, infrarouge, etc. . .).
- **L'unité d'alimentation** : élément primordial de l'architecture du capteur, c'est elle qui fournit en énergie toutes les autres unités. Elle correspond le plus souvent à une batterie ou une pile alimentant le capteur, dont les ressources limitées en font une problématique propre à ce type de réseau. La réalisation récente d'unité d'alimentation à base de panneaux solaires tente d'apporter une solution pour prolonger sa durée de vie.

3.5.Smaer_ dust :

Micromachines avec réseau sans fil et capteur (température, lumière...). Elles S'organisent automatiquement en réseau lorsque proches les unes des autres et elle utilise le système d'exploitions TinyOS conçu pour les réseaux de capteur sans fil.

Les Smaer_dust sont commercialisées depuis 2002. [92]

4. Conclusion :

L'informatique ubiquitaire est dotée d'un arsenal de technologie et d'équipement, qui lui permet de répondre aux besoins des utilisateurs dans différents domaines de la vie quotidienne.

1. **Conclusion générale**

   De nos jours, les applications informatiques doivent permettre à leurs utilisateurs de consulter des données n'importe quand, n'importe où, à travers leurs dispositifs d'accès. C'est l'idée sous-jacente de l'*informatique ubiquitaire* telle qu'elle a été décrite par Mark Weiser.

   Dans ce mémoire, nous avons présenté l'informatique ubiquitaire, son importance, ses caractéristiques, les technologies et les équipements utilisés.

2. **Perspectives**

L'informatique ubiquitaire ouvre plusieurs perspectives de recherche, que nous listons ci-dessous dessus :

a) Au niveau des applications :

- **Domaine de la maintenance**
  - Une nouvelle catégorie d'applications ubiquitaires, celle consacrée à la maintenance industrielle, dans le futur, elle constituera le gros gisement d'avantages concurrentiels encore inexploré. Une des principales composantes participant à la recherche de la qualité totale et à la réduction des coûts, la maintenance industrielle est devenue une fonction stratégique dans toutes les entreprises spécialisées dans la production, la fabrication et l'assemblage, le secteur aéronautique en est un bon exemple où la réparation d'avion à distance
- **Domaine de l'automobile**

  Dans les années avenir, nous assisterons à la fabrication de véhicule intelligent doté des capacités suivantes :

  - détection et évitement d'obstacles,
  - reconnaissance de balises,
  - assistance aux usagers,
  - contrôle à distance avec retour vidéo, etc.
  - doté de la capacité de communication avec d'autres machines (voiture, train et **panneaux de signalisation, …)**

    Une nouvelle notion apparaisse dans le monde les **automobiles autonomes**

- **Domaine militaire**

Dans le futur, on aura pas besoin d'humains sur le champ de bataille car ils seront remplacés par une armé de machines (des robots soldats). On estime que les premières armes entièrement autonomes seront prêtes d'ici 20 à 30 ans.

- **Domaine de la gestion des catastrophes naturelles :**

Dans les prochaines années, les équipes de sauvetage auront la capacité de la localisation des personnes et assurer une meilleure prise en charge grâce à au système d'information géographique entièrement automatisés.

- **Domaine administratif**

Dans le futur, nous aurons plus besoin ni d'un passeport biométrique, ni d'un badge d'accès car tous cela sera remplacé par un code-barres biométrique à la tête de l'être humain.

b) **Au niveau de développements logiciels :**

Les objets de la vie quotidienne deviennent des supports d'interaction entre les technologies développées et les humains Les dispositifs d'interaction se déversaient par leur forme et leur n'alite. Ces dispositifs utilisent des techniques intelligentes d'interaction impliquant la participation des gestes, tactile multiple, la commande vocale, le contrôle du regard, etc. De nos jours, la flexibilité et la plasticité des IHM deviennent une nécessité avec l'émergence de nouvelles technologies et d'interaction qui peuvent s'intégrer dans l'environnement ambiant. La sensibilité au contexte peut être utilisée pour adapter des interfaces homme-machine. Elle reste une piste très intéressante pour les chercheurs a fin de construire des interfaces plastiques, des interfaces adaptées au contexte.

- **L'invisibilité :** Il ne faut pas encombrer l'utilisateur avec des considérations qui ne le concernent pas [10]. Il faut qu'il se concentre sur la tache a réaliser [52]. En effet, avec l'augmentation du nombre de dispositifs informatises autour des utilisateurs [8], on assiste aussi a une augmentation du nombre d'applications qu'ils doivent gérer.
- **Gestion du Contexte :** Les systèmes ubiquitaires doivent être sensibles à leur environnement Physique pour prendre les décisions appropriées. Adoptent un modèle architecturelle a base de nouvelles technologies (infrastructure réseaux et dispositifs communicant). [4] ont approfondi la description de l'informatique ubiquitaire, Selon leur opinion, les capteurs fournissent des informations sur le contexte des systèmes ubiquitaires, ce qui rend ces derniers différents des systèmes classiques. Ils proposent un modèle ubiquitaire qui inclut quatre domaines : les dispositifs, les réseaux, les

intergiciels, les applications qui a enrichi la vision de Weiser en faisant une étude des possibilités offertes par l'informatique ubiquitaire et des recherches de ce domaine explique que l'informatique ubiquitaire se base sur les systèmes distribues et sur l'informatique mobile, en ajoutant des caractéristiques comme les espaces intelligents et l'invisibilité. Dans ses travaux initiaux présents un modèle centre utilisateur. Les auteurs illustrent l'importance des espaces intelligents dans lesquels le monde numérique et physique est lies d'une façon naturelle et transparente pour l'utilisateur.

Les travaux [15] intègrent le domaine des interfaces intelligentes qui permet aux usagers de contrôler et interagir avec des objets de manière intuitive. Enfin [6] abordent la sécurité, qui est un enjeu prépondérant dans la construction d'environnement pervasif.

- **Utilisation d'autres modèles au niveau intergiciels** dans le futur, Il serait très intéressant de prendre en compte plusieurs modèles différents au niveau intergiciel (par exemple basés sur CORBA, le pair-à-pair, etc.). Cela permettrait de choisir, parmi la gamme de modèles disponibles, les plus adaptés à l'application (lors de sa conception) ou même au contexte (dynamiquement lors de l'exécution du système). Cela apporterait une grande flexibilité pour ces intergiciels et leurs permettraient d'atteindre un degré d'adaptabilité plus important.

- **L'ingénierie des besoins ubiquitaires** : Elle représente aussi une thématique intéressante à développer pour améliorer la gestion des projets ubiquitaires. Jusqu'à présent les SI ubiquitaires ne bénéficient pas d'approches d'ingénierie des besoins qui leurs sont dédiées et qui permettent de se concentrer sur l'identification des caractéristiques ubiquitaires attendues d'un système. La démarche ECAR présenté ns le chapitre 2 c'est juste une introduction à ce thème qui est traité par une approche intentionnelle.

Il reste un travail de recherche important à effectuer par les spécialistes d'ingénierie des besoins. Ainsi, il est nécessaire de tester différents prototypes d'applications ubiquitaires auprès d'utilisateurs de différentes catégories sociales et professionnelles pour identifier les attentes et les prises de positions par rapport à ces systèmes. Il est possible d'ailleurs d'identifier de nouvelles fonctions ubiquitaires et de nouveaux besoins d'usages.

Finalement, les méthodes Agiles connaissent aujourd'hui un succès dans la réalisation des projets de développement. Nous considérons ce domaine combiné avec le développement des applications ubiquitaires comme une perspective importante de recherche. En effet, il est possible de combiner les pratiques Agiles dans notre démarche pour construire une vision

générale sur le déroulement du projet et faire une estimation temporelle de chaque phase et de l'organisation des différentes itérations.

- **Autonomie de la sensibilité au contexte :** L'autonomie d'un système représente les capacités d'un système à se reconfigurer de lui même. L'autonomie est principalement appliquée à l'administration de système pour laquelle il est important de minimiser le cout des interventions humaines de reconfiguration. Je pense qu'il serait intéressant d'appliquer les principes de l'autonomie a la sensibilité au contexte.

L'objectif est de permettre des reconfigurations de la sensibilité au contexte en fonction d'un certain nombre de facteurs :

- une base de connaissance qui s'enrichirait avec le temps sur les sensibilités au contexte acceptables dans tels ou tels environnements, des objets communicants accessibles ou non,
- des profils utilisateurs qui acceptent ou non tel type de reconfigurations.
- Evaluer des stratégies de modification du modèle de sensibilité au contexte a l'exécution en fonction de ce qui est découvert dans l'environnement ambiant permettrait d'ouvrir la voie à de la sensibilité au contexte autonome.

La sensibilité s'appuierait sur des bases de connaissances des types d'adaptations avec leur contexte d'utilisation (reconfigurations structurelles, comportementales) ainsi que de patrons d'adaptations et de leur contexte d'utilisation. Ces bases de connaissances devraient ˆêtre confortées par des études sur les usages des adaptations en environnement ubiquitaire.

– **Ingénierie dirigée par les modèles (IDM) pour les besoin ubiquitaire :**

Dans le domaine du génie logiciel, des intergiciels et de l'ubiquitaire, l'IDM est devenu incontournable. En effet, l'ubiquitaire demande à ce que le logiciel fonctionne sur des plates-formes matérielles variées. Or, il n'y a pas de convergence aujourd'hui, ni au niveau des langages, ni au niveau des systèmes disponibles sur les plates formes mobiles.

A partir d'un modèle applicatif (de sa structure et de son comportement), l'IDM permet grâce a des transformations de modèles de produire du logiciel `a destination d'un large spectre de matériel. Par ailleurs, comme nous l'avons montre dans ce document, les modèles spécifiques, a plus haut niveau d'abstraction, sont utilisables par les intergiciels pour, d'une part, exprimer une préoccupation dans un modèle conforme a un méta-modèle spécifique et, d'autre part, gérer cette préoccupation.

Ces raisons nous confortent dans la conviction que des travaux de recherche sur les environnements de développement de logiciel couplés avec les fonctions de transformation et les services des intergiciels facilitent la production de logiciel pour l'ubiquitaire

La Gestion d'énergie : Les technologies d'accès radio possèdent des modèles de consommation d'énergie différente. Ces différences sont intrinsèquement liées aux technologies utilisées (Wifi, réseaux mobiles). Pour un terminal mobile qui se base sur une batterie de faibles capacités ce qui peut le rendre indisponible pour un temps indéfini. Cet aspect est primordial puis qu'il affecte pervasive. Plus ces éléments sont nombreux, plus l'intégration devient complexe. Il faut prendre en compte des aspects tels que la qualité de service, la sécurité et la fiabilité.

Sécurité et confidentialité :

La sécurité reste un de défis majeur de l'informatique ubiquitaire :

- L'environnement pervasif est très ouvert permettant l'accès au système informatique aux personnes présentes dans cet espace.
- L'accès à certains équipements ou aux données personnelles des utilisateurs doit être fortement sécurisé.

La mise en place d'un système de sécurité nécessite le développement des aspects suivants :

- **L'authentification :**

L'identification d'un utilisateur donnée se fait a travers un processus d'authentification. L'évolution technologique offre une variété d'outils (code PIN, login, carte bancaire, badge, empreinte digitale, empreinte rétinienne, reconnaissance vocale).

- **Le Contrôle d'accès :**

Dans les environnements pervasifs l'information est accessible n' importe où et n'importe quand. De ce fait, l'administration et l'intégration des différentes politiques de sécurité deviennent plus complexes dans la mesure où ces dernières sont hétérogènes d'un point de vue structurel (rôle) et sémantique (codification, langue).

- **La confidentialité et la protection de la vie privée** :

L'utilisation de la nouvelle technologie a mauvais escient, peut porter atteinte à la vie privée des usagers. Un utilisateur a une perception très limitée des risques potentiels issus des différents équipements embarqué.

Références bibliographies

1] Article ziad neham dans Techniques de l'Ingénieur 2003

[2]Cliquet, G. 2007. « Informatique Ambiante: Nouveaux défis pour le Design». In Recherche 2000 . <<http://turing.lecolededesign.com/~gcliquet/>>.

[3] WEISER (M.). . The Computer for the 21st Century. Scientific American 1991.

[4] Amazon.com. amazon web services, 2012. http://aws.amazon.com/.

[5] Gitlevich, V., Evans, E., Hu, Y., and Nilsson, J. Domain-driven design community, 2012. Case Studies, http://domaindrivendesign.org.

[6] Stéphane Lavirotte, Gaëtan Rey, Jean-Yves Tigli « Introduction à l'Informatique Sensible au Contexte », 2009.

[7] Satyanarayanan, M. « Ubiquitous Computing : les challenges logiciels ». IEEE Personnal communication, (August 2001)

[8] Want, R., A. Hopper, V. Falcao et J. Gibbons. 1992 a. « The Active Badge Location System». ACM Transcation on Information Systems vol.

[9] Dey, A. K., et G. D. Abowd. 2000. « Towards a Better Understanding of Context and Context-Awareness ». In Workshop on the What, Who, Where, When, and How of Context Awareness Conference on Human Factors in Computer Systems (CHI2000). The Hague, Netherlands

[10] Sadeh, N. M., F. L. Gandon et O. B. Kwon. 2005. « Ambient Intelligence: The MyCampus Experience ». Carnegie MellonUniversity, Technical Report CMU-ISRI-05-123,(July 2005).

[11] Saha, D., et A. Mukherjee. « Pervasive computing: a paradigm for the 21st century ». IEEE Computer journal, (Mars 2003)

[12] Julien Mercadal Résolution de Approche langage au développement logiciel : application au domaine des systèmes d'informatique ubiquitaire thèse de doctorat à l'Université de Bordeaux.

[13] Journal du CHU de Nice. Accueil des Urgences le parcours du patient optimisé. Http://www.chu-nice.fr/site_CHU/file/site/magazines/MAG_39.pdf, 2006.

[14] M. Jimenez, F. Rosique, P. Sanchez, B. Alvarez et A. Iborra. Habitation : A Domain-Specific Language for Home Automation. IEEE Software,. IEEE Computer Society Press, 2009

[15] Laforest, F. « Systèmes d'informations pervasifs », 2007.

[16] D. Saha et A. Mukherjee. Pervasive Computing: a Paradigm for the 21st Century. IEEE Computer,. IEEE Computer Society Press, 2003.

[17] A. Schmidt. Ubiquitous Computing – Computing in Context. Thèse de doctorat, Lancaster University, 2002.

[18] Laforest, F. « Systèmes d'information pervasifs », 2007.

[19] Mohand Boughanem, Lynda Tamine-Lechani, JoséMartinez, Sylvie Calabretto, Jean-Pierre Chevallet, Un nouveau passage à l'échelle en recherche d'information, publié dans "Ingénierie des Systèmes d'Information (ISI) .8 Feb 2009.

[20] A. K. Dey, G. D. Abowd et D. Salber. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction, L. Erlbaum Associates Inc., 2001.

[21] K. Henricksen et J. Indulska. Developing Context-Aware Pervasive Computing Applications : Models and Approach. Pervasive and Mobile Computing, 2(1):37–64. Elsevier Science Publishers B. V., 2006.

[22] F. Zhu, M. W. Mutka et L. M. Ni. Service Discovery in Pervasive Computing Environments. IEEE Pervasive Computing,. IEEE Educational Activities Department, 2005

[23] W. K. Edwards et R. E. Grinter. At Home with Ubiquitous Computing: Seven Challenges. Dans Proceedings of the 3rd International Conference on Ubiquitous Computing (UbiComp'01) , Atlanta, Georgia, USA. Springer-Verlag, 2001

[24] A. D. Birrell et B. J. Nelson. Implementing Remote Procedure Calls. ACM Transactions on Computer Systems, ACM Press, 1984.

[25] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi et S. Weerawarana. Unraveling the Web Services Web: an Introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing. IEEE Educational Activities Department, 2002.

[26] R. Grimm. One.world : Experiences with a Pervasive Computing Architecture. IEEE Pervasive Computing, 3(3):22–30. IEEE Educational Activities Department, 2004.

[27] R. Grimm, J. Davis, E. Lemar, A. Macbeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble et D. Wetherall. System Support for Pervasive Applications. ACM Transactions on Computer Systems, 22(4): ACM Press, 2004.

[28] J. P. Sousa et D. Garlan. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. Dans Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture (WICSA'02), . Kluwer, B.V., 2002.

[29] S. Staab et R. Studer. Handbook on Ontologies. Springer-Verlag, 2009

[30] M. Jimenez, F. Rosique, P. Sanchez, B. Alvarez et A. Iborra. Habitation : A Domain-Specific Language for Home Automation. IEEE Software. IEEE Computer Society Press, 2009

[31] G. Agha. Actors: a Model of Concurrent Computation in Distributed Systems. MIT Press, 1986.

[32] P. Barron et V. Cahill. YABS: a Domain-Specific Language for Pervasive Computing based on Stigmergy. Dans Proceedings of the 5th International Conference on Generative Programming and Component Engineering (GPCE'06), Portland, Oregon, USA. ACM Press, 2006.

[33] H. R. Schmidtke et W. Woo. Towards Ontology-Based Formal Verification Methods for Context Aware Systems. Dans Proceedings of the 7th International Conference on Pervasive Computing (PERVASIVE'09), Nara, Japan. Springer-Verlag, 2009.

[34] A. Ranganathan et R. H. Campbell. Provably Correct Pervasive Computing Environments. Dans Proceedings of the 6th IEEE International Conference On Pervasive Computing and Communications (PerCom'08), IEEE Computer Society Press, 2008.

[35] Theodore C. Belding. The distributed genetic algorithm revisited. In Proceedings of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc 1995

[36] Object Management Group. « CORBA components ». Adopted Specificationformal/02-06-65, OMG, juin 2002. Version 3.0.

[37] Object Management Group. Common object request broker architecture (corba/iiop). http://www.omg.org/spec/CORBA/3.1, January 2008.

[39] Didier Hoareau THESE Doctorat de l'université de Bretagne Sud thème Composants ubiquitaires pour réseaux dynamiques décembres 2007

[41] R. Grimm. « System Support for Pervasive Applications ». Thèse de doctorat, Université de Washington, décembre 2002.

[42] D. Garlan, D. Siewiorek, A. Smalagic, et P. Steenkiste. « Project Aura: towards

Distraction free pervasive computing ». IEEE Pervasive Computing, special issue on Integrated Pervasive Computing Environments, avril 2002.

[43] M. Román et R. H. Campbell. « Gaia : enabling active spaces ». Dans ACM SIGOPS European Workshop , 2000.

[44] J. Dedecker, T. Van Cutsem, S. Mostinckx, T. D'Hondt et W. De Meuter. Ambient Oriented Programming in AmbientTalk. Dans Proceedings of the 20th European Conference on Object-Oriented Programming (ECOOP'06), pages 230–254, Nantes, France. Springer-Verlag, 2006.

[45] P. Barron et V. Cahill. YABS: a Domain-Specific Language for Pervasive Computing based on Stigmergy. Dans Proceedings of the 5th International Conference on Generative Programming and Component Engineering (GPCE'06) , Portland, Oregon, USA. ACM Press, 2006.

[46] M. Jimenez, F. Rosique, P. Sanchez, B. Alvarez et A. Iborra. Habitation : A Domain-Specific Language for Home Automation. IEEE Software, 26(4):30–38. IEEE Computer Society Press, 2009.

[47] S. R. Ponnekanti, B. Johanson, E. Kiciman et A. Fox. Portability, Extensibility and Robustness in iROS. Dans Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom'03), IEEE Computer Society Press, 2003.

[48] R. Grimm, J. Davis, E. Lemar, A. MacBeth, S. Swanson, T. E. Anderson, B. N. Bershad, G. Borriello, S. D. Gribble, et D. Wetherall. « System support for pervasive applications ». ACM Trans. Comput. Syst.2004.

[49] N. Medvidovic, R. N. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages. IEEE Transactions on Software Engineering.

[50] R. N. Taylor, N. Medvidovic et E. M. Dashofy. Software Architecture: Foundations, Theory, and Practice. John Wiley & Sons,

[51] Henricksen, K. et Indulska, J. (2006). Developing context-aware pervasive computing applications: models and approach. Pervasive and Mobile Computing, 2006.

[52] Achilleos, A., Yang, K. ET Georgalas, N. (2010). Context modeling and a context-aware framework for pervasive service creation: A model driven approach. Pervasive and Mobile Computing 2010

[53] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord et J. Stafford. Documenting Software Architectures: Views and Beyond (2nd ed.). Addison-Wesley, 2010.

[54] Saha, A. Mukherjee. Pervasive Computing: A Paradigm for the 21st Century. IEEE Computer Society, 36(2):25-31. 2003.

[55] Ben Cheikh, A., Front, A., Giraudin, J.-P. ET Coulondre, S. (2012). An engineering method for context-aware and reactive systems. In Sixth International Conference on Research Challenges in Information Science, RCIS, Valence, Espagne. IEEE. Best Paper Award

[56] Vieira, V., Tedesco, P. ET Salgado, A. (2010). Designing context sensitive systems : An integrated approach. In Expert Systems with Applications. Elsevier

[57] Garlan, D., and Shaw, M. An introduction to software architecture. Advances in software engineering and knowledge engineering 1, January 1993

[58] Shaw, M., and Garlan, D. Software Architecture : Perspectives on an Emerging Discipline. Prentice Hall, 1996.

[59] Booch, G., Jacobson, I., and Rumbaugh, J. The Unified Software Development Process. Prentice Hall, 1999.

[60] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. Pattern-Oriented Software Architecture Volume 1 : A System of Patterns. Buschmann, 1996.

[61] Perry, D. E., andWolf, A. L. Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes 17, 4 (Oct. 1992)

[62] Bass, L., Clements, P., and Kazman, R. Software Architecture in Practice (2nd Edition). Addison-Wesley Professional, 2003.

[63] M. Caporuscio, A. Carzaniga et A. L. Wolf. Design and Evaluation of a Support Service for Mobile, Wireless Publish/Subscribe Applications. IEEE Transactions on Software Engineering. IEEE Computer Society Press, 2003.

[64] Baresi, L., and Ghezzi, C. The disappearing boundary between development-time and run-time. In Proceedings of the FSE/SDP workshop on Future of software engineering research (New York, NY, USA, 2010).

[65] IEEE. Recommended practice for architectural description of software-intensive systems, 2007. IEEE ISO/IEC 42010 :2007.

[67] Garlan, D. An introduction to the aesop system, 1995. The ABLE Project.

[68] Medvidovic, N., and Taylor, R. A classification and comparison framework for soft-ware architecture description languages. IEEE Transactions on Software Engineering (2000)

[69] Shaw, M., DeLine, R., Klein, D. V., Ross, T. L., Young, D. M., and Zelesnik, G. Abstractions for software architecture and tools to support them. IEEE Trans. Softw. Eng. 21, 4 (Apr. 1995), 314–335.

[70] Vestal, S. Metah. SIGSOFT Softw. Eng. Notes 25, 1 (Jan. 2000),

[71] Hoare, C. A. R. Communicating sequential processes. Commun ACM 21, 8 (Aug. 1978)

[72] Taylor, R. N., Medvidovic, N., Anderson, K. M., Whitehead, Jr., E. J., and Robbins, J. E. A component- and message-based architectural style for gui software. In Proceedings of the 17th international conference on Software engineering (New York, NY, USA, 1995),

[73] Parnas, D. L. On the criteria t o be used in decomposing systems into modules. Communications of the ACM 15, 12 (Dec. 1972),

[74] Michelson, B. M. (2006). Event-driven architecture overview: Event-driven soa is just part of the eda story. Rapport technique, Seybold Group.

[75] Michlmayr, A., Rosenberg, F., Leitner, P. et Dustdar, S. (2008). Advanced event processing and notifications in service runtime environments. In Distributed Event Based Systems, Rome, Italy. ACM.

[76] Pietzuch, P. R. (2004). Hermes: A Scalable Event-Based Middleware. These de doctoral, Queens' College University of Cambridge.

[77] Cilia, M., Bornhovd, C. ET Buchmann, A. (2001). Moving active functionality from centralized to open distributed heterogeneous environments. In et al., C. B., éditeur : CoopIS'01

[78] Sharon, G. et Etzion, O. (2008). Event processing network model and implementation. Rapport technique, IBM.

[79] Lilia Gzara, "Les patrons pour l'ingénierie des systèmes d'information Produit Thèse pour obtenir le grade de Docteur, Institut National Polytechnique de Grenoble.

[80] Dounia Mansouri, "Architecture orientée objet de type SIAD pour L'élaboration de contenus pédagogiques", département d'informatique, 2002

[81] Yann GRIPAY, Jean-Marc PIERSON, Charles-Eric PIGEOT, Vasile-Marian SCUTURICI Laboratoire LIRIS Une architecture pervasive sécurisée : PerSE 2005

[82] Hong J., Landay J.A., An architecture for privacy-sensitive ubiquitous computing. In Proceedings of MobiSYS '04: ACM Press, 2004.

[83] Grimm R. et al., System Support for Pervasive Applications, ACM Transactions on Computer Systems, Vol. 22, No. 4, November 2004,

[84] M. Y. I. Idris, Y. Y. Leng, E. M. Tamil, N. M. Noor, Z. Razak, "Car Park System: A Review of Smart Parking System and its Technology".Information Technology Journal, Vol. 8 (2), 2009.

[85] Mark W., « Turning pervasive computing into mediated spaces », IBM Systems Journal, vol , 1999,

[86] Ferscha, A. Holzmann, C. Oppl , S. (2004). Context awareness for group interaction support. international workshop on Mobility management and wireless access protocols, ACM Press New York, N Y, USA,

[87] UMTS de Javier Sanchez et Mamadou Thioune Hermes Sciences - Lavoisier Paris 2008 (3e édition)

[88] Alix C. & Kodron C. (2003), Coopérer, se comprendre, se rencontrer, Paris- Berlin : Office franco-allemand pour la jeunesse (Documents de travail de l'OFAJ)

[89] Beuscart, R. et Grave, C. et Bricoteau, D. et Purro, N. (1993). Les étapes de définition d'un système d'information hospitalier: la place des utilisateurs. Informatique et santé, Vol.6,

[90] Lavirotte, S. Lingrand, D. Tigli, J. (2005). A propos du contexte et des différentes méthodes de sélection associées. Rapport de recherche ISRN I3S/RR-2005-06-FR,

[91] M. Weiser. The computer for the 21st century. Scientific American, vol. 265, 1991.

[92] Nigay, L. (1994). Conception et Modélisation Logicielle des Systèmes Interactifs : Application aux Interfaces Multimodales. Thèse de l'Université Joseph Fourier, Grenoble.

[93] Naismith, L. Lonsdale, P. Vavoula, G. Sharples, M. (2004) literature Review in Mobile Technologies and Learning. Futurelab series,

[94] http://www.atelier.net/trends/articles/mobiles-se-transforment-reseau-de-capteurs

[95] Messaoud Belloula ; « La géolocalisation dans les réseaux de capteurs sans fil ; Etude de cas : Utilisation en agriculture ».Thèse de Magistère ; Université Hadj Lakhder-Batna.

[96] Boudjaadar Amina ; « Plateforme basée Agents pour l'aide à la conception et la simulation des réseaux de capteurs sans fil ». Thèse de Magistère ; Université de Skikda ; 2009/2010