État de l'art, proposition et mise en oeuvre d'un intergiciel pour la communication dans un environnement réel augmenté entre acteurs mobiles

Abdelhnine Mahfoudi

CONSERVATOIRE NATIONAL DES ARTS ET METIERS
CENTRE REGIONAL DE LYON

Mémoire

présenté en vue d'obtenir le

Diplôme d'ingénieur CNAM en informatique

Par

Abdelhnine MAHFOUDI

Etat de l'art, proposition et mise en œuvre d'un intergiciel pour la communication dans un environnement réel augmenté entre acteurs mobiles

---

Soutenu le 17 septembre 2010

JURY Président : M. Christophe PICOULEAU
Membres : M. Bertrand DAVID
M. René CHALON
M. Claude GENIER
M. Daniel MULLER

Remerciements

Tout d'abord, je tiens à remercier le professeur Bertrand David d'avoir accepté d'encadrer mon mémoire au sein de l'équipe du laboratoire LIESP qu'il dirige à l'école centrale de Lyon.

Je remercie également René Chalon, Maître de conférences pour les discussions riches et variées.

Mes remerciements à David Bain, auditeur du CNAM de la promotion 2009, avec qui j'ai eu l'heur de passer un moment trop court.

Mes remerciements vont aussi aux membres des équipes du service Communication, du service de la Valorisation, des Partenariats, et du Contrôleur de gestion pour tous les moments partagés soit à la pose café, soit durant le déjeuner ou les arcanes du fonctionnement de l'Ecole Centrale de Lyon étaient dévoilés.

Je remercie aussi mes enfants et mon épouse pour la patience enduré par mon absence occasionnelle et circonstanciée.

Enfin, je remercie mes parents et particulièrement ma mère qui m'a toujours prodigué des conseils et des paroles chaleureuses qui continuent à jamais, de faire effet.

Table des matières

Table des illustrations :

Liste des tableaux :

CHAPITRE I

I. INTRODUCTION

Avec l'évolution actuelle de l'informatique, nous sommes amenés à utiliser de plus en plus des dispositifs hétérogènes, qui interagissent entre eux en vue de coopérer.

Ces dispositifs doivent ainsi communiquer grâce à des connexions réseau réagissant à l'activité de l'utilisateur de façons passive ou active. La façon passive est par exemple, le déclenchement d'un capteur lors du passage de l'utilisateur sans que celui-ci ne le fasse de façon volontaire. L'autre façon est à l'inverse, par un acte volontaire de l'utilisateur en vue, par exemple, d'obtenir des informations. Quelques soient les façons, le résultat des actions permet d'enrichir l'interface de l'utilisateur en apportant des informations supplémentaires issues du contexte.

On parle alors d'environnement réel augmenté, qui est géré par un système informatique en conjuguant les données informatiques contenues dans le système avec l'apport des données externes, et ceci en temps réel. De plus, les personnes qui sont amenées à se mouvoir peuvent aussi communiquer entre elles dans une cadre collaboratif.

C'est ainsi qu'apparaissent de nouvelles problématiques inhérentes à ce type d'environnements mobiles. En effet, aussi bien les architectures logicielles que les réseaux sont différents de ceux habituellement rencontrés. La différence provient des ressources matérielles comme la mémoire, la puissance du microprocesseur, l'utilisation des ondes au niveau du réseau qui d'une façon générale est moins gourmande en ressources. L'inconvénient de cette technologie se situe au niveau des liaisons qui ont la particularité d'être plus intermittentes avec des débits variables, mettant à mal la notion de continuité de service. Le réseau constitue de ce point de vue une ressource à part entière dont découlent la conception et la réalisation d'applications.

Notre sujet recoupe les domaines de la Réalité Augmentée et l'Informatique Ambiante. Ces deux disciplines sont relativement proches et traitent toutes les deux de l'aspect virtuel dans le but d'apporter une plus value au monde réel. Une de leur différence est le fait que la Réalité Augmentée met l'utilisateur au centre, en lui apportant directement les informations supplémentaires alors que l'Informatique Ambiante repose davantage sur une informatique disséminée au travers de l'environnement. Ce sont deux domaines de l'informatique qui sont relativement récent mais qui sont prometteurs. Le premier

écrit traitant de la Réalité Augmentée (angl. Augmented Reality) est celui intitulé : "Computer-Augmented Environments : Back to the Real World" [Wellner 1993]. On peut ainsi définir, la Réalité Augmentée comme un système commandé par l'informatique qui apporte des informations supplémentaires à celles de l'environnement et ceci en temps réel. On peut également rencontrer le terme de "Réalité Mixte" (mixage de la réalité et du virtuel) pour définir le même phénomène. Les informations dit virtuelles peuvent être d'ordre visuel, sonore ou kinesthésique et enrichissent (ou pourrait dire augmentent) le monde réel. Les applications touchent des domaines variés comme les jeux vidéo, les jeux d'éducation, l'industrie du cinéma, le domaine médical, la maintenance…

En ce qui concerne l'Informatique Ambiante, il existe d'autres termes pour la désigner. On parle aussi d'informatique ubiquitaire (angl. Ubiquitous), diffuse (angl. Pervasive), sensible au contexte (angl. context-aware). La référence académique est celle de Mark Weiser qui a été le premier à l'avoir défini [Weiser 1991]. Ce qui caractérise l'informatique ambiante est le fait qu'elle soit omniprésente jusqu'à faire partie du moindre objet de la vie de tous les jours. On peut la retrouver dans des appareils informatiques comme les portables, les PDA, les TabletPC, les téléphones portables mais demain on pourra la retrouver dans un grand nombre d'objets comme une tasse, une peluche, un tableau qui seront muni d'un dispositif RFID par exemple. Le champ d'application est suffisamment large pour imaginer que d'autres dispositifs ou objets futurs trouveront toute leur place dans des domaines comme les loisirs ou dans un but de faciliter la vie. Les domaines ne sont pour l'instant que partiellement identifiés mais n'attendent qu'à être défrichés et sont d'ores et déjà prometteurs.

Notre travail s'inscrit dans le domaine de recherche de l'équipe du laboratoire LIESP de l'Ecole Centrale de Lyon. Le compte-rendu est structuré de la façon suivante : Un état de l'art sur les principaux aspects concernant l'environnement réel augmenté (démarches, formalismes, réseaux et intergiciels avec une place conséquente à la technologie sans fil ZigBee) constitue le contenu du prochain chapitre. Le chapitre suivant est consacré à la conception et la réalisation couvant les aspects réseau et intergiciel. Avant la conclusion, le dernier chapitre décrit deux études de cas qui ont été menées.

CHAPITRE II

II. ETAT DE L'ART

II.1. ETUDE, DEMARCHES, METHODES ET FORMALISME

Nous commençons par la mise en place d'un cadre théorique avec l'adoption de la démarche MDA avant l'application des modèles de l'informatique des systèmes coopératifs mobiles.

II.1.1. MDA

L'architecture dirigée par les modèles ou *MDA* (Model Driven Architecture) [MDA 2001] est une démarche de réalisation de logiciel (dans notre cas de middleware) proposée par l'*OMG* (Object Management Group). La démarche consiste à l'élaboration de différents modèles en débutant par le modèle métier : *CIM* (Computation Independent Model), suivi du modèle *PIM* (Platform Independent Model) et pour terminer par *PSM* (Platform Specific Model). La figure ci-dessous (fig. 1) illustre ce fait, et correspond ainsi, à la séparation des modèles métier, technologique et d'implémentation.

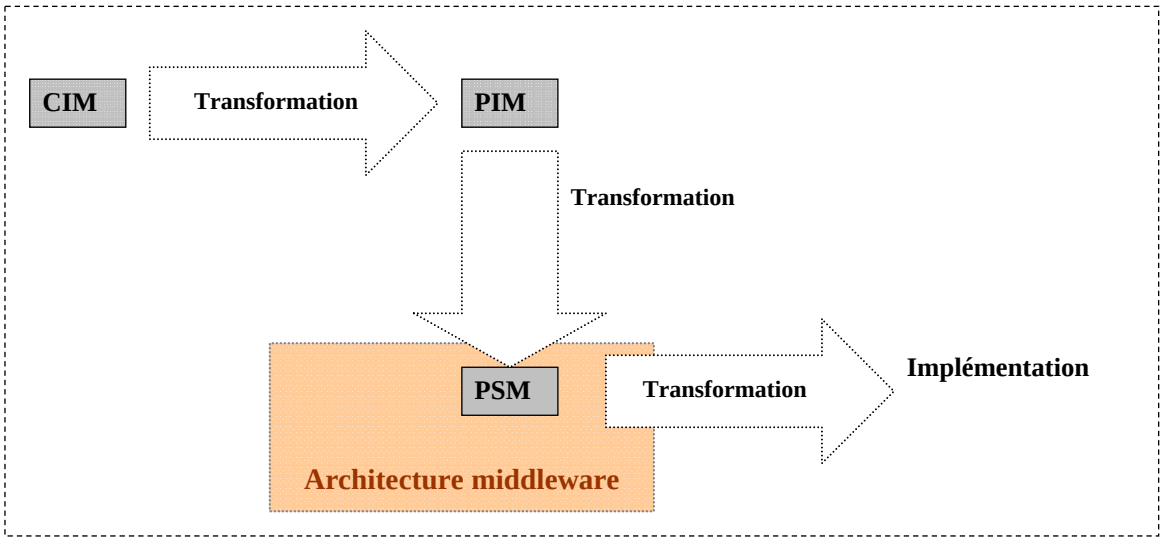

Fig. 1 : Etape de transformation avec MDA.

II.1.1.1. Modèle CIM (Computation Independent Model)

Le modèle CIM, dit métier, correspond aux besoins et à l'environnement de l'utilisateur et est indépendant de l'informatisation. Ce modèle peut être exprimé dans le formalisme UML, par des diagrammes de cas d'utilisations.

II.1.1.2. Modèle PIM (Platform Independent Model)

Le modèle *PIM* concerne la conception théorique du système informatique. En effet, à cette étape, on ne se préoccupe surtout pas de l'implémentation du middleware et entre autres choses, du choix du langage, de la plateforme. On utilise plutôt les Design Patterns afin d'exprimer les meilleurs pratiques ou la réutilisabilité est respectée par l'utilisation de bibliothèques d'objets, le cas échéant. Ainsi, l'application est modélisée suivant le respect des fonctionnalités exprimées au niveau du CIM, sans toutefois le choix de la plateforme. Ce modèle correspond au diagramme de classe pour le formalisme UML.

II.1.1.3. Modèle PSM (Platform Specific Model)

Le modèle PSM correspond à la dernière étape concernant la transformation vers un modèle spécifique à la plateforme cible.

II.1.2. Méthodologie : CoCSys (Cooperative Capillary Systems)

La méthode CoCSys [Delotte 2006] qui a été élaborée dans le cadre du laboratoire LIESP est en adéquation avec notre sujet puisqu'elle traite de l'étude de systèmes coopératifs mobiles et collaboratifs.

C'est en effet, une démarche qui couvre tout le cycle de vie d'une application de ce type. On peut y rencontrer aussi bien des serveurs propres à l'informatique classique que des dispositifs plus légers comme les Tablette PC, PDA, lecteur RFID qu'on trouve eux, davantage dans un environnement informatique de *Réalité Augmentée* (et/ou ubiquitaire) et répond à notre problématique d'environnement réel augmenté. Cette méthode s'inscrit dans une démarche de type MDA (Model Driven Architecture) et se déroule en plusieurs phases (cf. fig. 2) que nous détaillons ci-dessous :

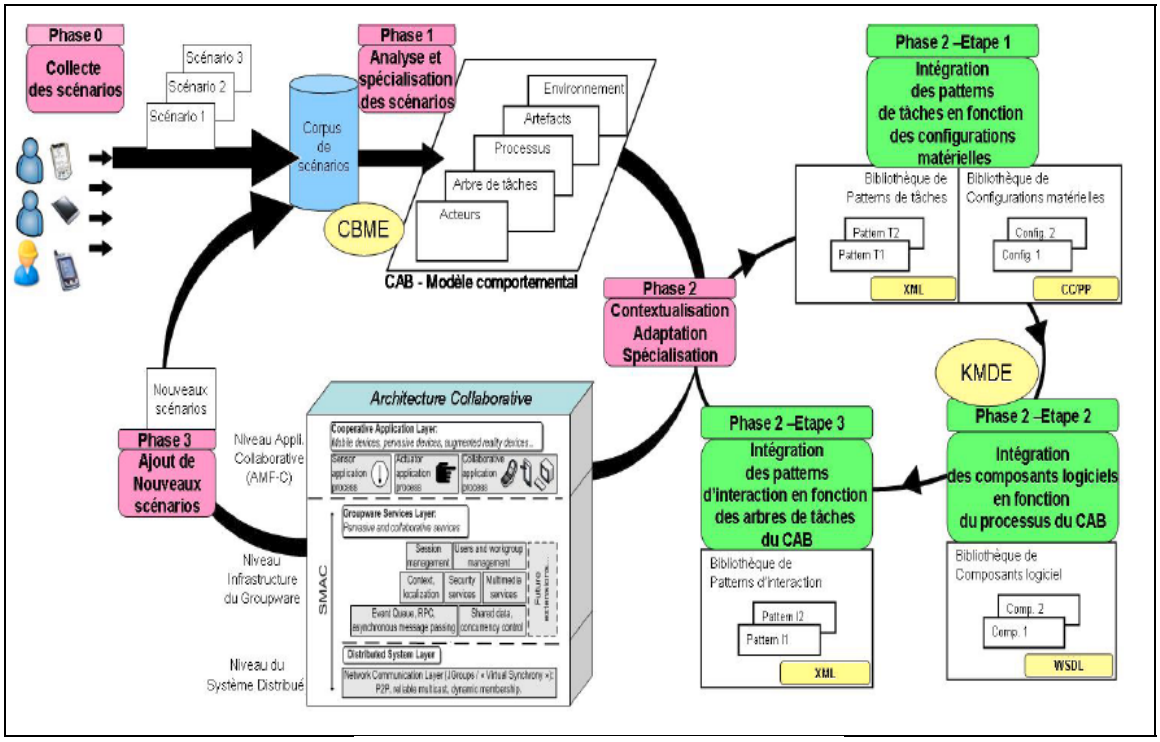

Fig. 2 : Processus CoCSys (d'après [Delotte 2006]).

- **phase 0** :

Dans un premier temps, on écrit de façon naturelle sous forme textuelle ou graphique, les besoins des utilisateurs. Ceci sert de support pour la suite des étapes.

  - Etape 1 : A cette étape, on élabore les scénarios contextualisés (cf. exemple fig. 3).
  - Etape 2 : Sert à enrichir les scénarios.
  - Etape 3 : Cette étape permet de faire apparaitre les buts du système.
  - Etape 4 : Permet d'identifier des scénarios manquants.

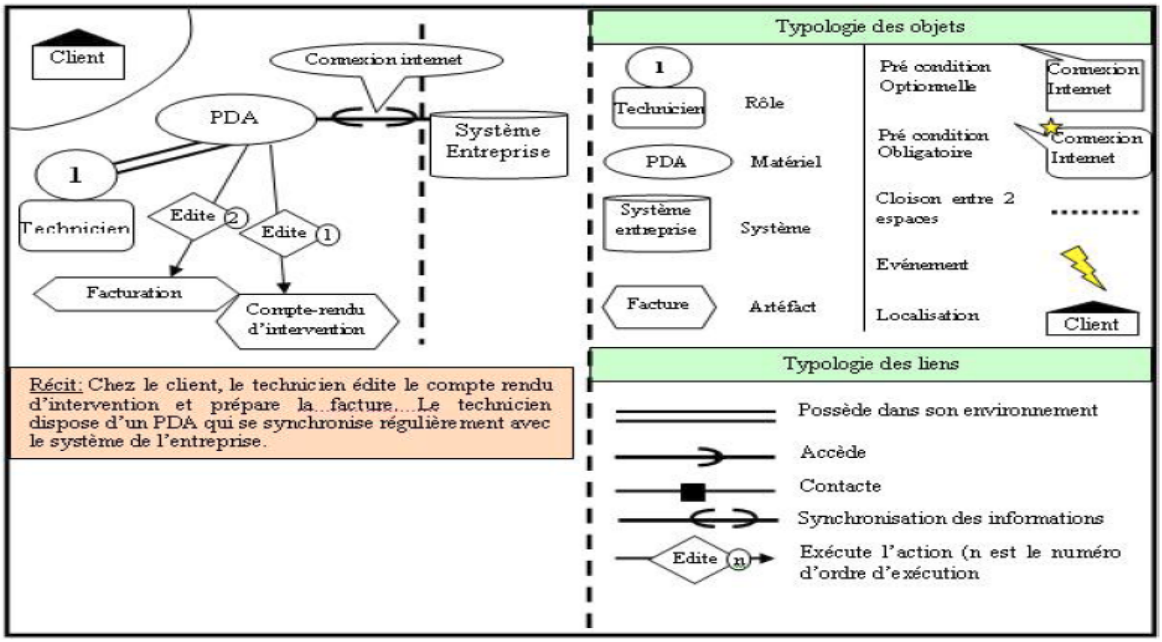

Fig. 3 : Exemple de scénario contextualisé (selon [Delotte 2006]).

- **phase 1** : Transformation, des divers scénarios en modèle coopératif de comportement collaboratif (*CAB* : Collaborative Application Behavior). Cette phase se fait en trois étapes : préparation des informations, construction du modèle et vérification. L'objectif est la production d'un ensemble de modèles faisant apparaitre les activités et les caractéristiques du système (cf. fig. 4).

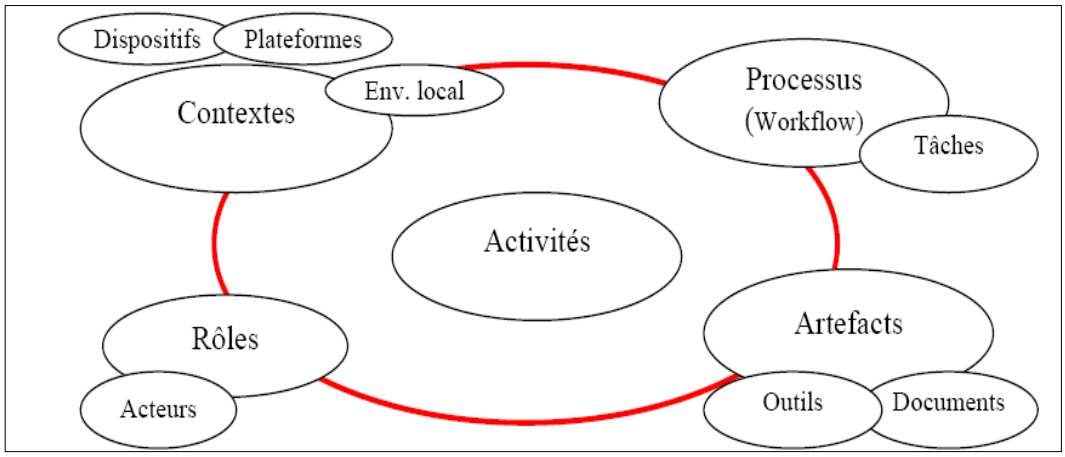

Fig. 4 : Informations synthétisées dans le modèle CAB (d'après [Delotte 2006]).

- **phase 2** : Adaptation et création de l'application collaborative. Il s'agit précisément de transformer le modèle CAB en une implémentation spécifique qui répond à l'exigence du modèle d'architecture collaborative comme représenté à la figure 5.

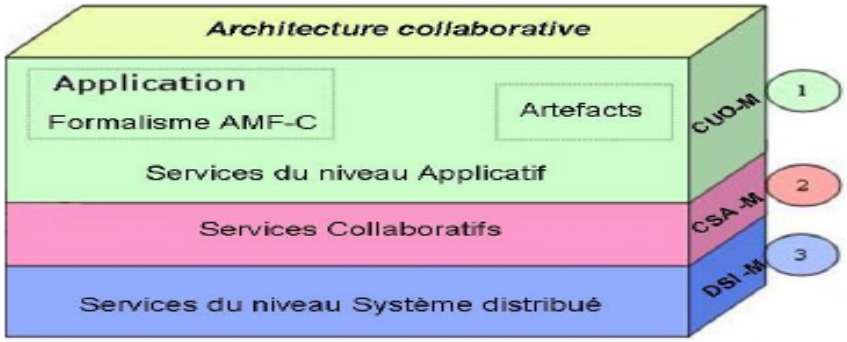

Fig. 5 : Architecture à 3 niveaux (d'après [Delotte 2006]).

Chaque couche représente une fonctionnalité particulière comme décrite ci-dessous :

- Le niveau 1 : **CUO-M** (Collaboration User-Oriented Model) représente la transposition des éléments du modèle comportemental en services applicatifs.
- Le niveau 2 : **CSA-M** (Collaborative System Architecture Model) représente les services collaboratifs qui ont la particularité d'être génériques comme les contrôles de sessions, d'utilisateurs et de groupes.
- Le niveau 3 : **DSI-M** (Distributed System Infrastructure Model) correspond aux services système proposés par les couches basses de la plateforme.

- **phase 3** : Dans cette phase, on formalise les évolutions voulues par les utilisateurs.

II.1.3. IRVO (Interacting with Real and Virtual Objects)

II.1.3.1. Presentations

IRVO est une méthodologie et un formalisme de conception des systèmes de la réalité mixte et collaboratifs développé par René Chalon [Chalon 2004] au laboratoire LIESP de l'ECL. Elle se situe comme une extension de la démarche CoCSys, en proposant une modélisation fine des interactions entre les utilisateurs et le système de *réalité mixte*. Elle modélise plus particulièrement l'interaction au niveau d'une tâche élémentaire et de ce fait, on l'utilise avec l'arbre de tâches. On peut ainsi construire une modélisation globale de l'application.

En effet, le modèle de tâche permet d'analyser et de représenter l'activité utilisateur sous la forme d'une structure de tâches qui décrit ce que l'utilisateur peut ou doit faire. Lorsqu'il représente les tâches supportées par un système, on parle d'une utilisation à une fin descriptive alors qu'on parle d'une utilisation à une fin prescriptive pour un système en cours de développement.

II.1.3.2. Représentation graphique

La représentation est relativement simple et met en lumière le monde réel et virtuel ainsi que les principales entités aux nombres de trois que nous présentons dans les paragraphes suivants.

II.1.3.2.1. Frontières

La frontière est une notion inhérente à IRVO afin de représenter des propriétés des entités de façon géographique lorsque les entités sont situées dans le monde réel (cf. repères ②③④⑤ de la fig. 6) et de distinguer également le Monde réel du virtuel (cf. repère ① de la fig. 6).

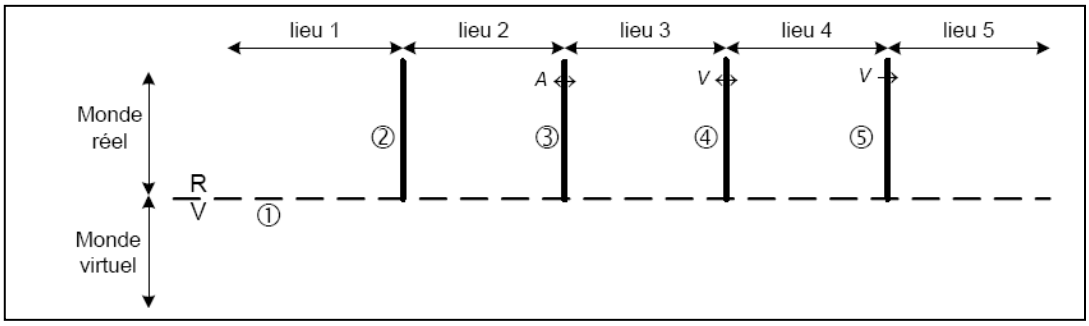

Fig. 6 : Représentation des frontières, selon [Chalon 2004].

Nous distinguons avec une granularité plus fine, les différents cas du Monde réel.

- En effet, lorsque la frontière est hermétique entre deux lieux comme le lieu 1 et le lieu 2 (item ②de la figure 6) la représentation se fait uniquement avec un trait vertical continu.
- Pour le cas entre les lieux 2 et 3 (item ③ de la fig. 6), on a un franchissement partiel, à double sens par un flux audio (représentation par flèche bidirectionnelle avec "A").
- Pour le cas entre les lieux 3 et 4 (item ④ de la fig. 6), c'est aussi un franchissement partiel de part et d'autre de façon visuelle (flèche bidirectionnelle avec "V").
- Le cas entre les lieux 4 et 5 (item ⑤ de la fig. 6), peut correspondre à une personne se situant au niveau du lieu 4 qui visualise un objet se situant sur le lieu 5 (flèche unidirectionnelle avec "V").

II.1.3.2.2. Utilisateurs

Une des catégories du modèle IRVO est l'utilisateur qui est identifié par la lettre "**U"** et possède les caractéristiques détaillées ci-dessous à la figure 7 :

- le nom,
- des attributs de mobilité,
- des canaux de communications représentant trois sens :
  - Le canal kinesthésique/haptique avec la lettre **KH**

- Le canal Visuel avec la lettre **V**
- Le canal audio avec la lettre **A**

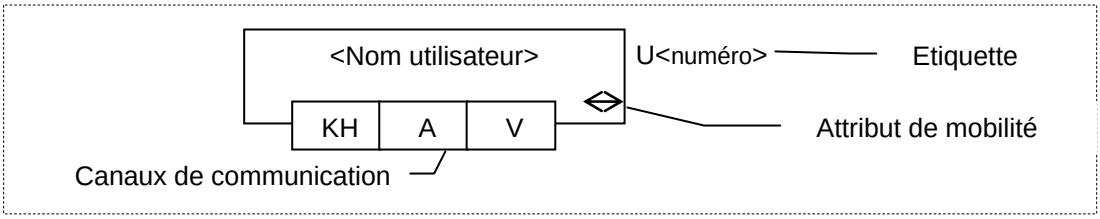

Fig. 7 : Représentation d'un utilisateur.

II.1.3.2.3. Objets

Les objets sont les seuls entités qui peuvent être réelles ou virtuelles. De ce fait, on les représente par un rectangle accompagné au coin en haut-droit du rectangle de l'étiquette "O" pour objet et "T" pour outil. Pour le cas d'entité virtuelle, on rajoute un "v", et concernant l'entité réelle, on rajoute un "r" (cf. exemples fig. 8a et 8b). De plus, une partie d'un objet peut se partager sur le monde réel alors que l'autre partie est dans le monde fixe. On parle dans ce cas d'objet mixte (cf. fig. 8 c).

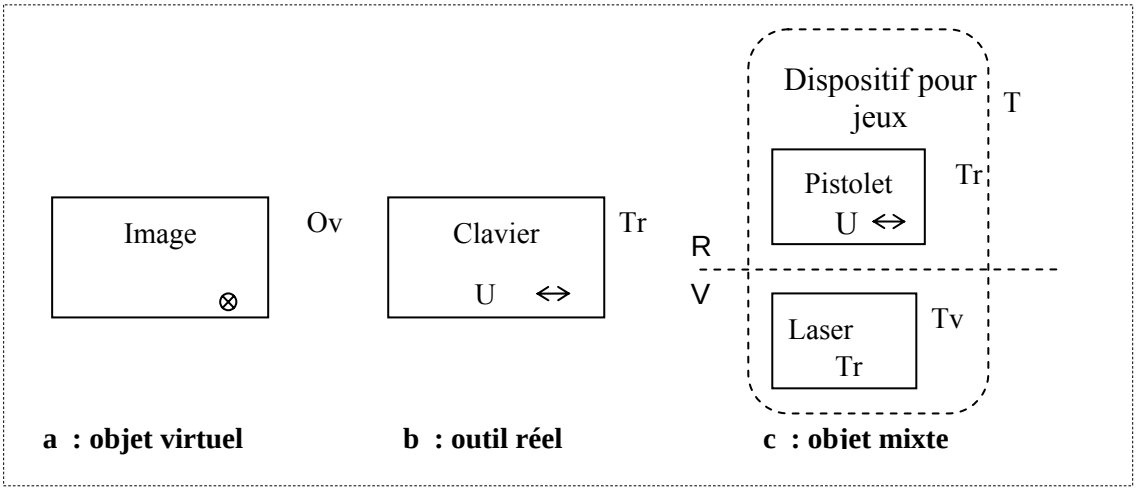

Fig. 8 : Exemple de représentation d'objet

II.1.3.2.4. Modèle interne

Le modèle interne noté M (cf. fig. 9) sert à représenter le modèle comportemental du logiciel et à gérer les objets (et outils) virtuels. Il est en quelque sorte, responsable de l'augmentation des objets et de ce fait, gère le travail collaboratif.

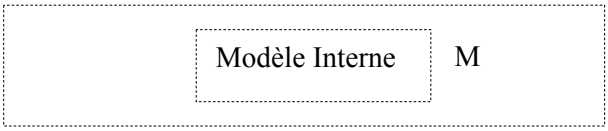

Fig. 9 : Représentation du Modèle Interne.

II.1.3.2.5. Transducteurs

Les transducteurs sont des dispositifs positionnés à la frontière du monde réel et virtuel afin de faire transiter l'information dans un sens ou dans l'autre.

- Lorsque l'information est transformée du monde réel vers le monde virtuel pour être transformé en données numériques, l'opération est réalisée par le senseur (cf. fig. 10a).
- Lorsque l'information provient du monde virtuel sous forme numérique pour être transformé vers le monde réel de façon physique, l'opération est assuré par l'effecteur (cf. fig. 10b).

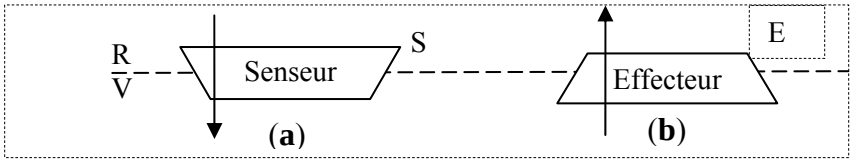

Fig. 10 : Représentation des transducteurs ([Chalon 2004]).

II.1.3.2.6. Les relations entre entités

Les relations sont représentées par des flèches afin de traduire les échanges d'informations entre entités. Elles peuvent également être accompagnées de textes pour apporter des informations supplémentaires. Les relations de moindre importance sont représentées par des pointillés contrairement aux relations principales qui sont matérialisées avec des traits pleins.

Les différentes relations sont détaillées ci-dessous :

- relation de l'utilisateur vers objet (ou outil) (cf. fig. 11 a).
- relation entre utilisateurs (cf. fig. 11 b). On distingue deux sous cas :
  - Un seul des deux utilisateurs est actif. Cela peut correspondre comme dans le cas ① fig. 11b à l'action physique de l'utilisateur 2 sur l'utilisateur 1. Bien que cet exemple traite du canal kinesthésique/haptique, on peut rencontrer d'autre cas correspondant à l'utilisation des autres canaux, voire une combinaison d'utilisation des différents canaux.
  - Les deux utilisateurs sont actifs. Dans l'exemple ② de la fig. 11b, les utilisateurs se regardent. Les autres canaux peuvent également être sollicités.

- relation entre objets (et/ou outil).

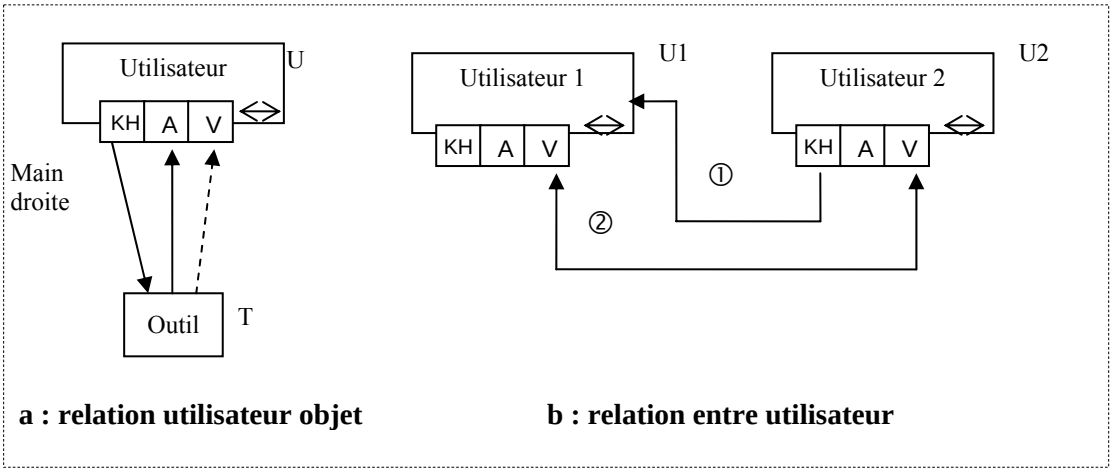

Fig. 11 : Exemple de représentation de relations

- Fusion de relations : la représentation de l'augmentation d'un objet mixte se fait par le symbole : ⊕traduisant l'union de sa partie virtuelle et de sa partie réelle (cf. fig. 12).

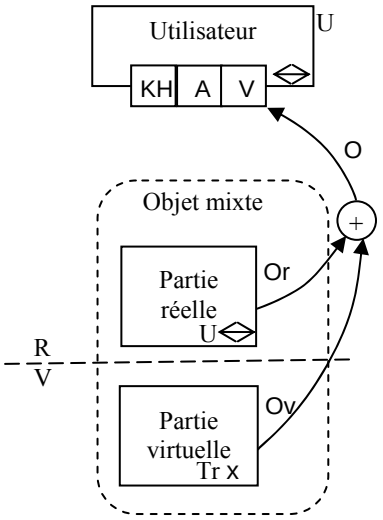

Fig. 12 : Exemple de représentation d'une fusion de relations.

II.1.3.2.7. Mobilité des entités

La représentation se fait par un symbole en bas à droite de l'entité (cf. exemple fig. 13) suivant la signification du tableau de la fig. 14.

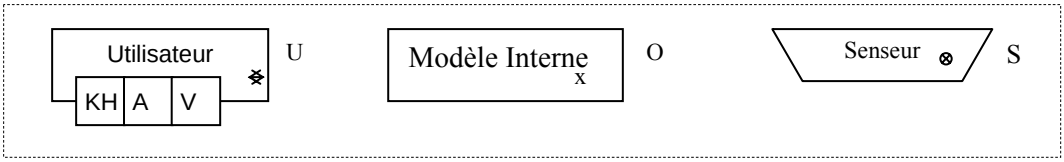

Fig. 13 : Exemples d'entité avec des symboles de mobilité.

Symbole	Signification
⟷	L'entité est mobile durant l'exécution de la tâche
x	L'entité est immobile durant la durée de la tâche
⊗	L'entité est immobile durant l'exécution de l'application

Fig. 14 : Symboles et significations des propriétés de mobilité.

II.1.4. Conclusion

MDA est une démarche propre au génie logiciel qui permet de construire des logiciels en s'appuyant sur une succession de modèles. Ces modèles sont d'abord le modèle métier : CIM (cf. § II.1.1) suivi du modèle PIM et enfin suivi du modèle PSM. La méthodologie CoCSys s'applique plus particulièrement à la conception des systèmes mobiles et la méthode IRVO permet de concevoir des systèmes propres à la Réalité Mixte. Ces deux méthodes s'appliquent particulièrement bien à notre problématique de systèmes collaboratifs capillaires.

L'autre aspect de notre problématique est la mobilité où les utilisateurs peuvent être amenés à collaborer à travers des interfaces comme celles de PDA, nous amène à nous y intéressé. En effet, un des besoins est de connaitre des informations du niveau des couches réseau comme l'adresse IP pour être renseigné à travers l'interface utilisateur.

II.2. ETUDE DE L'ASPECT RESEAU

Un réseau informatique est défini comme un ensemble de matériels et de logiciels qui sont mis en œuvre pour assurer les communications entre les différents "terminaux". Ici, on considère le terme terminal dans le sens premier du terme à savoir une machine ou un équipement qui se situe au bout de l'échange d'information [Pujolle 3Edit]. Dans le cas qui nous intéresse à savoir la réalité mixte, ce pourrait être par exemple, une sonde.
Aussi, la classification des réseaux se fait suivant plusieurs critères qui sont : leur type de transmissions, leur taille, leur performance dont nous détaillons ci-dessous les particularités.

- En ce qui concerne le premier critère : **type de transmission**, la distinction se fait suivant le support comme le filaire, l'optique, le sans fil et suivant le mode de transmission (commutation 1 à 1).

- Pour le second critère : **taille**, on peut noter quatre niveaux différents.

  - Le premier : *PAN* (Personal Area Network) correspond au réseau personnel reliant des appareils électroniques. La distance qui sépare les éléments les plus éloignés de ce réseau varie de 10 cm à 10 m pour un débit de 56 Kb/s à 1 Mb/s .
  - Le second : *LAN* (Local Area Network) correspond au réseau local reliant des ordinateurs d'un même bâtiment. La distance varie de 1m à 2 km. Le débit peut varier de 10 Mb/s pour un réseau éthernet à 1 Gb/s pour un réseau Gigabit éthernet.
  - Le troisième : *MAN* (Metropolitan Area Network) correspond au réseau métropolitain dont la taille se situe à l'échelle d'une ville. La distance varie de 1 m jusqu'à une centaine de kilomètre. Le débit varie de quelques centaines de Kb/s à 20 Mb/s.
  - Le quatrième : *WAN* (Wide Area Network) pour le réseau à grande échelle reliant des ordinateurs se situant sur des continents différents à l'instar d'internet par exemple. La distance dépasse la centaine de kilomètres. Les débits sont relativement élevés et résulte d'un compromis entre la distance, le coût et la bande passante.

- Pour le troisième critère : **performance**, on distingue les réseaux suivant leur bande passante qui concerne le débit et le délai qui mesure le temps écoulé entre l'émission d'un bit et sa réception (sa latence).

Concernant les besoins en mobilité propre à notre projet, nous avons opté pour le réseau mobile sans fils afin d'assurer la communication entre plusieurs acteurs mobiles. Ce type de réseau nécessite ainsi l'existence d'un intergiciel qui assure l'aspect inhérent à la réalité augmentée. Le choix se porte plus particulièrement sur le type de réseau dit ad hoc que nous présentons au paragraphe suivant.

II.2.1. Réseaux sans fil

II.2.1.1. Réseau ad hoc

Dans le langage courant, la locution latine : "ad-hoc" signifie : "*pour cela*" ou "*dans un but précis*" (synonyme d'adéquat) et en ce qui concerne l'informatique, un *réseau ad hoc* est un réseau sans fil capable de s'organiser sans infrastructure définie à priori [Pujolle 2008]. A l'origine, ce type de réseau à fait l'objet d'un intérêt de la part des militaires pour lequel, ils voyaient un avantage évident à disposer d'un réseau capable de s'auto-configurer et s'auto réparer avec des pertes d'éléments comme les nœuds ou les liens et ce, quelque soit le champ militaire et la nature des opérations. Depuis ce type de réseau s'est répandu dans le domaine civil, avec des applications au niveau domotique, médical, secteur automobile.

On parle de "réseau ad hoc mobile" (MANET : Mobile Ad hoc NETwork) pour tout système autonome dynamique composé de nœuds mobiles interconnectés par des liens sans fils.

Ces réseaux auto-organisés sont formés à partir d'entités mobiles communicantes sans nécessité d'infrastructure au préalable. Les entités mobiles peuvent être constituées d'éléments hétéroclites comme des ordinateurs portables, des TabletPC, des assistants électroniques, des capteurs, et par conséquent, ces éléments ne présentent pas tous les mêmes capacités de puissances de calculs et de stockage. Le fait que les acteurs se déplacent de façon aléatoire et arbitraire a pour conséquence un comportement du réseau fortement dynamique dans le temps et l'espace.

Un nœud mobile peut communiquer directement à un autre nœud s'il se situe à porté de transmission et relaye les messages en jouant le rôle de routeur. Ainsi, deux nœuds qui ne se trouvent pas dans le même voisinage peuvent bénéficier de la transmission de nœuds intermédiaires grâce à la communication dite par multi-sauts.

Un réseau peut être autonome ou connecté à une infrastructure fixe, ce qui lui donne l'avantage de remplir une plus large gamme d'application. Il possède les mêmes propriétés et problèmes liés au réseau sans fils. Le canal radio peut alors, être limité en termes de capacité, il peut aussi être plus exposé aux pertes et être sujet à des variations dans le temps. Les liaisons sans fils ont aussi l'inconvénient d'être asymétriques et souvent non sécurisées.

Il existe deux protocoles de routage ad-hoc normalisé par le groupe MANET : le protocole OLSR (*Optimized Link State Routing)* et le protocole AODV (*Ad-hoc On-demand Distance Vector*).

Le premier protocole correspond à un type proactif. En effet, le meilleur chemin est déterminé par des relais multipoints (ou MPR, *MultiPoint Relay*). Ces relais correspondent à des nœuds importants qui informent de l'état des liens et évitent ainsi les messages de supervision.

Le second est plutôt de type réactif. Il gère en effet les routages unicast et multicast. Les inconvénients de ce protocole proviennent du temps et du trafic généré pour la mise en place des routes. L'avantage d'AODV est de ne pas générer de trafic en l'absence d'envoi d'information.

II.2.1.2. Réseau mesh

Les réseaux mesh [Pujolle 2008] sont des réseaux ad-hoc pour lesquels les points de routage restent immobiles. Ces points de routage sont reliés entre eux afin de constituer un réseau sans fil et font office de points d'accès à des utilisateurs mobiles. Aussi on peut couvrir une large zone géographique sans générer les travaux importants propres à la pose de câbles.

II.2.1.3. Réseau de capteurs

Un réseau de capteurs [Pujolle 2008] est formé de capteurs qui émettent et reçoivent des informations entre eux. Il possède des propriétés spécifiques par rapport aux architectures classiques. En effet, la conséquence de la miniaturisation des capteurs entraine des problèmes de communication et plus particulièrement de routage, de contrôle des erreurs et de gestion de l'énergie électrique.

Ce type de réseaux forme des réseaux mesh avec l'utilisation de protocoles spécifiques pour assurer une faible consommation d'énergie comme le protocole simplifié d'AODV. De façon générale, les protocoles sont proposés par l'IETF (*Internet Engineering Task Force*) et ne prennent pas en compte les aspects optionnels.

Les réseaux de capteurs concernent Zigbee que nous examinons plus en détail au paragraphe suivant.

II.2.2. Protocole ZigBee

L'intérêt d'un protocole est d'apporter des règles de fonctionnement lors d'une communication.

Parmi les réseaux ad-hoc nous avons choisi de nous intéressé particulièrement au protocole ZigBee qui permet de transmettre des données entre des équipements périphériques sur une faible distance avec une moindre consommation électrique. Ci-après, nous présentons les différentes couches et le fonctionnement de ZigBee.

II.2.2.1. Architecture protocolaire de ZigBee

ZigBee est un standard de communication sans fil, basé sur la norme IEEE 802.15.4 pour les couches physique et liaison (cf. fig. 15). Il a été formalisé par un groupement d'industriel : l'Alliance ZigBee.

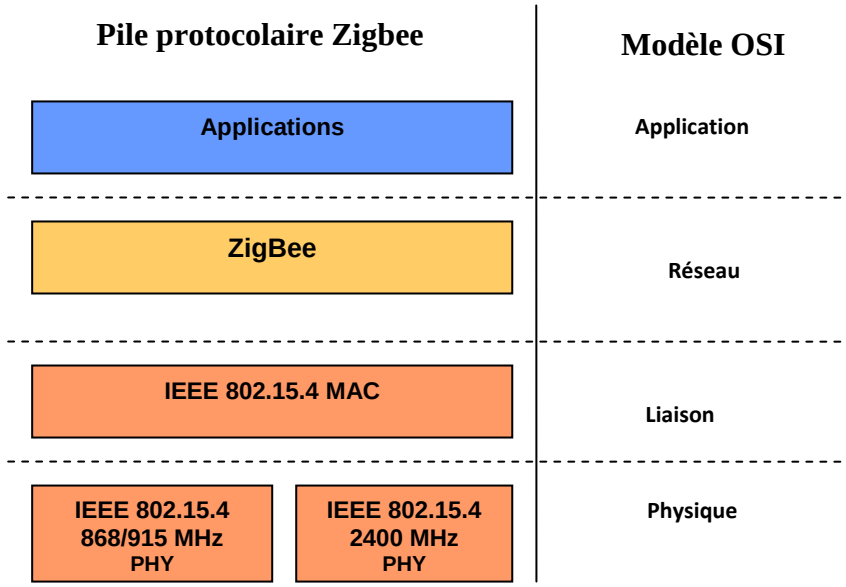

Fig. 15 : Pile protocolaire ZigBee

La norme IEEE 802.15.4 est prévue pour les réseaux à dimension personnelle sans fil : *WPAN* (Wireless Personal Area Network), ce qui, bien entendu nous intéresse en tant qu'utilisateur du protocole ZigBee. Nous détaillons les deux couches basses dans les paragraphes suivants.

II.2.2.2. Couche physique IEEE 802.15.4

La couche physique de ce protocole gère les aspects :

- d'émission et réception des paquets au travers du canal radio,
- d'activation et désactivation de l'interface radio,
- de sélection du canal radio,
- mise en œuvre du protocole d'accès CSMA après évaluation du canal.

Les caractéristiques de cette couche sont :

- 3 bandes de fréquences ISM (Industrial, Scientific and Medical) (cf. fig.16) avec les spécificités suivantes :
  - 868 MHz avec 1 canal.
    - ◊ 1 bit par symbole ce qui fait un débit symbole de 20 kBaud.
    - ◊ Modulation BPSK.
    - ◊ Pour chaque symbole, on a une correspondance d'une séquence "15 chips" pour étalement DSSS ce qui fait 300 kchips/s.

- 915 MHz avec 10 canaux.
  - ◊ 4 bits par symbole ce qui fait un débit symbole de 62,5 kBaud.
  - ◊ 1 bit par symbole, ce qui fait un débit symbole de 40 kBaud.
  - ◊ Modulation BPSK.
  - ◊ Pour chaque symbole, on a une correspondance d'une séquence "15 chips" pour étalement DSSS ce qui fait 600 kchips/s.

- 2.4 GHz avec 16 canaux. Les caractéristiques de cette bande sont :
  - ◊ 4 bits par symbole soit un débit symboles de 62,5 kBaud.
  - ◊ Modulation orthogonale O-QPSK avec 16 symboles.
  - ◊ Correspondance à chaque symbole d'une séquence d'étalement "32 chips" pour un étalement DSSS.

- Débits de 20kb/s (pour la 864 MHz), 40 kb/s (pour la 915 MHz) et 250 kb/s (pour la 2.4 GHz).
- Type de méthode d'accès au support : CSMA-CA (Carrier Sense Multiple Access – Collision Avoidance). La technique CSMA consiste à écouter le canal avant une émission. Lorsque la détection des collisions n'est pas possible comme dans le cas des réseaux sans fil, on utilise alors la méthode CSMA-CA. Cette méthode se base en effet, sur l'utilisation d'accusés de réception et de temporisateurs lors de la communication.
- Protocole fiable avec acquittement.
- Faible consommation.

	Bande	Usage	Débit	Nombre De canaux
868 MHz	ISM	Europe	20 kb/s	1
915 MHz	ISM	Amérique	40 kb/s	10
2.4 GHz	ISM	Mondial	250 Kb/s	16

Fig. 16 : Caractéristiques des bandes de fréquences ZigBee.

II.2.2.3. Sous-couche MAC IEEE 802.15.4

Les caractéristiques de cette couche sont :

- L'adressage de trames réalisé en 64 bits ou en 16 bits pour un adressage court,
- gestion des balises,
- 3 niveaux de sécurité qui correspondent, soit à pas de sécurité, soit à liste ACL, soit à chiffrement AES 128.

Il existe deux modes d'accès au réseau :

- Mode non beacon (sans balise) qui utilise CSMA-CA pour l'accès au médium.
- Mode beacon (balise) : un coordinateur envoi périodiquement une balise qui délimite une "superframe". C'est le coordinateur qui fixe le format de cette

"superframe" et se compose de 16 intervalles de temps. Cela permet ainsi de synchroniser tous les dispositifs du réseau.

A la figure 17, nous avons représenté l'échange de données dans les cas de ces deux modes.

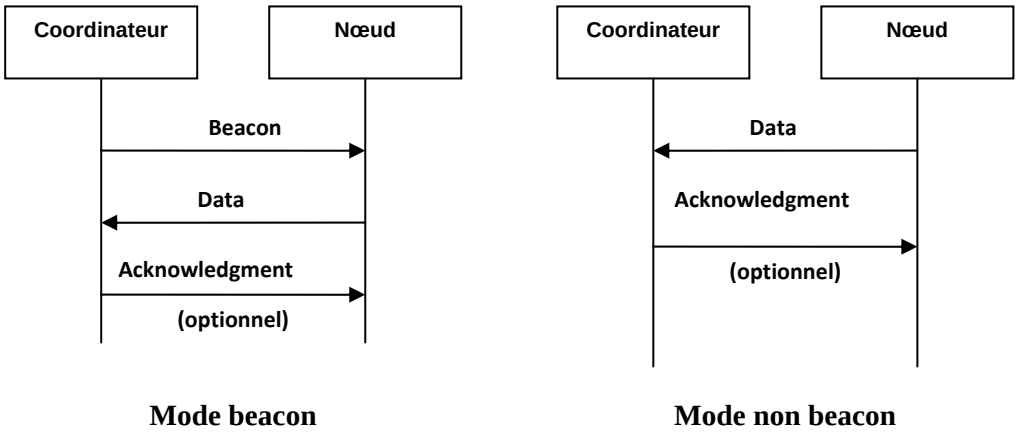

Fig. 17 : Echange de données.

La norme définit également trois types d'équipements qui sont :

- le coordinateur,
- l'équipement à fonctionnalités complètes : *FFD* (Full Function Device) qui peut correspondre aussi bien à un coordinateur, un routeur, ou à un équipement terminal (End Device) comme un capteur par exemple.
- l'équipement à fonctionnalités réduites : *RFD* (Reduced Function Device) peut correspondre à un équipement terminal muni de capteurs.

Pour la création d'un réseau, il faut prendre soin de vérifier la présence d'au moins un équipement FFD et des équipements RFD qui soient configurés pour utiliser le même canal radio.

II.2.2.4. Couche ZigBee

ZigBee apporte des fonctionnalités supplémentaires à la norme IEEE 802.15.4 se situant au niveau de la couche réseau. Ces fonctionnalités sont les suivantes :

- la topologie réseau de types : point à point, en étoile ou maillé (mesh) pour la couche réseau (cf. fig. 18).
- la sécurité par l'emploi optionnel d'un chiffrement AES 128 pour la couche présentation.
- la définition de profil d'utilisation pour la couche application.

En ce qui concerne l'aspect du routage, il y a deux cas à prendre en considération :

- soit direct : l'adresse est définie au niveau de la partie MAC de la trame.

- soit indirect : l'élément émetteur ne connait pas l'adresse du destinataire et c'est le coordinateur ou le routeur qui dirige l'information suivant sa table de routage.

La norme ZigBee préconise l'utilisation de l'algorithme AODV (Ad hoc On-Demande Vector routing) pour le cas de réseau maillé. On parle d'algorithme de routage réactif, c'est-à-dire que la route est établie après qu'une demande soit faite.

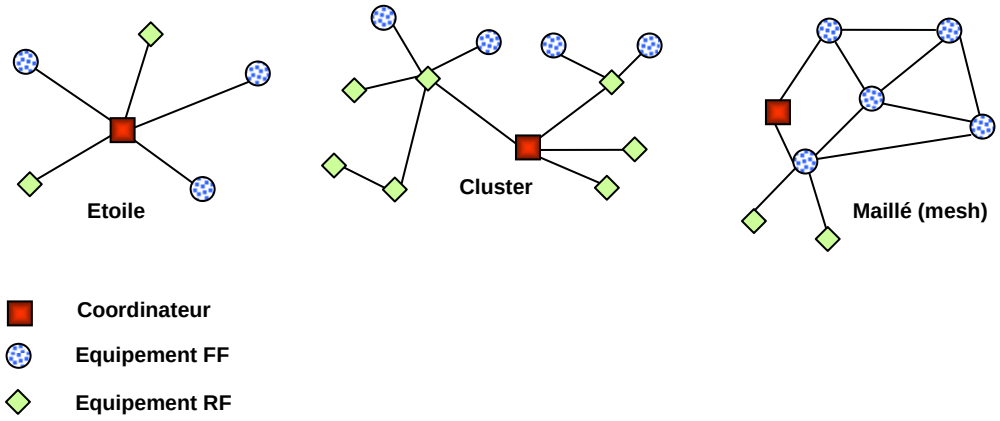

Fig. 18 : Topologies des réseaux ZigBee.

II.2.2.5. Conclusion

La technologie de communication sans fils a tendance à se développer avec la généralisation du téléphone portable mais aussi avec des objets munis d'interfaces appropriés comme les PDA (Personal Digital Assistant), les TabletPC, et de façon plus général à tout objet pouvant participer à "l'augmentation" d'un environnement. Par conséquent, elle nous intéresse car elle participe plus particulièrement à la communication d'acteurs mobiles.

Il existe plusieurs standards tels Wifi (IEE 802.11a/b/g/n/n-draft), Bluetooth (IEE 802.15.1) et ZigBee (IEE 802.15.4) (cf. fig. 19). Les deux premiers standards sont apparus avant le troisième et de ce fait, il existe beaucoup plus d'études et de réalisations les concernant. Ce dernier a en revanche des caractéristiques techniques particulières comme l'optimisation de l'utilisation du médium hertzien. C'est ainsi que le mode de fonctionnement "doze" ou somnolence permet le basculement en veille après chaque émission et réalise des économies d'énergie. Il est en effet, le moins gourmand en énergie. ZigBee peut assurer l'autonomie d'un dispositif avec une pile de 1,5 v qui peut se compter en années alors que pour la technologie Bluetooth, l'autonomie se compte en jours et pour la technologie Wifi c'est en heures. C'est ce qui explique son succès malgré l'antériorité des deux autres technologies.

Tout ceci explique et justifie pourquoi nous avons choisi ce protocole pour notre travail.

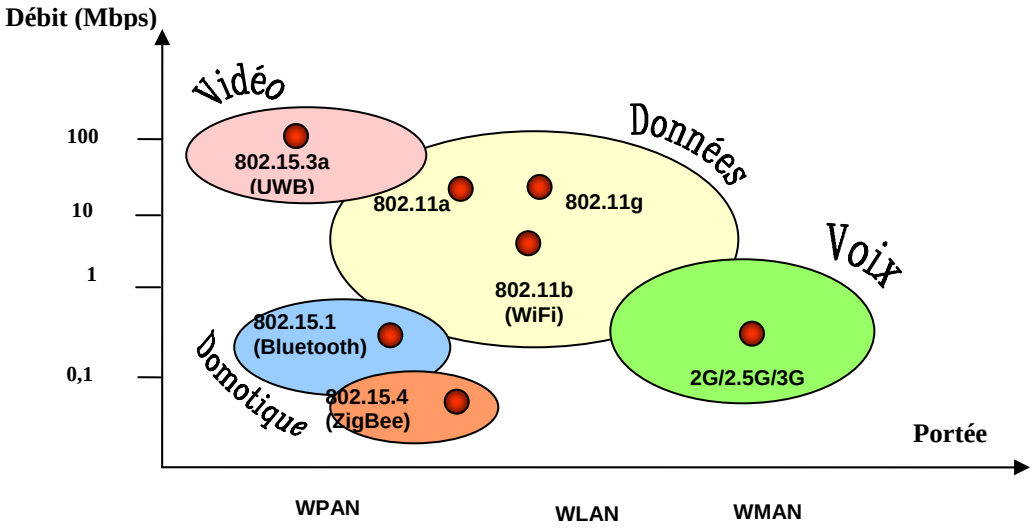

Fig. 19 : Positionnement de ZigBee dans les réseaux sans fil.

La faible utilisation d'énergie par la mise en veille périodique a des conséquences sur la continuité de service qui risque d'être mis à mal au niveau de l'utilisateur. Par la suite, nous allons vérifier, s'il n'existe pas de technique logiciel pour compenser cette caractéristique. A cette fin, nous allons étudiés dans le chapitre suivant, les paradigmes, les mécanismes et les outils qui conviennent à notre problématique dans la perspective de réaliser un middleware.

II.3. PRESENTATION ET ETUDE DES INTERGICIELS

Un intergiciel (angl. middleware) [Krakowiak 2003] est un logiciel qui assure un ensemble de fonctionnalités ou de services. Il occupe une couche intermédiaire (cf. fig. 20) entre, d'une part, l'application et d'autre part, le système d'exploitation et les protocoles de communication.

Depuis leur apparition vers les années 1990, les intergiciels ont pris une place importante dans le développement de l'informatique répartie. Ce mouvement ne se dément pas avec l'apparition et la progression de nouveaux domaines comme la réalité mixte, l'informatique ubiquitaire et les systèmes mobiles. De ce fait, les intergiciels évoluent et s'adaptent aux nouvelles contraintes. Aussi dans un environnement nomade, l'intergiciel doit permettre de construire des applications distribuées solides en masquant les détails de bas niveau.

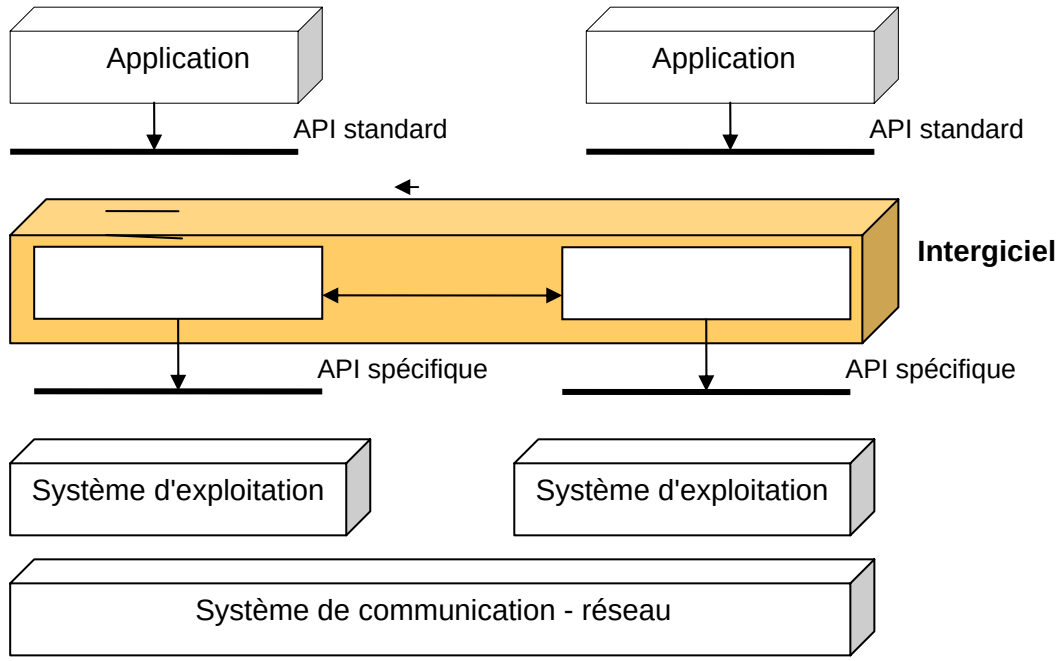

Fig. 20 : Représentation schématique d'un intergiciel

L'intergiciel a quatre fonctions principales :

1. Masquer la répartition du fait qu'une application s'exécute à des emplacements répartis, correspondant à des endroits géographiques distincts.
2. Masquer l'hétérogénéité propre à l'exécution d'application dépendant aussi bien du matériel, du système d'exploitation que du protocole de communication.
3. Offrir des interfaces normalisées et uniformes afin de faciliter la construction d'application ainsi que leur réutilisation, leur portage et leur interopérabilité.
4. Offrir des services communs afin de faciliter la coopération entre applications.

Pour satisfaire aux besoins de taille, d'adaptabilité, de flexibilité, d'utilisation, d'aspect distribué, de durée de vie, de modularité, d'adaptation, les paradigmes, composants et

services on été utilisés voire inventés pour la circonstance. Cela a entrainé, l'apparition de nombreux frameworks ainsi que des intergiciels fonctionnant avec ces technologies. Afin de choisir la technologie qui réponde au mieux à notre problématique, nous avons d'abord exploré les principes du modèle orienté composant puis dans un second temps celui orienté service et enfin nous terminons par le modèle composant orienté services.
En fonction de notre étude, nous avons fait un choix dans la perspective de réaliser une maquette.

II.3.1. Modèle et plateformes à base de composants

Il existe certes plusieurs définitions concernant les composants mais celle donnée par Szyperski [Szyperski 1996] est souvent citée comme référence de base et c'est à ce titre que nous l'utilisons nous aussi. On peut dire qu' "un composant est une unité de composition avec des interfaces contractualisées et un contexte de dépendance exprimé explicitement. Un composant peut être déployé de façon indépendante et est sujet à composition par une tierce personne".

Par conséquent, un composant est un regroupement de données et de traitements propre à ses données. En disant cela, on définit aussi ce qu'est un objet pour la programmation orienté objet. Ce qui n'est évidemment pas suffisant. On doit rajouter que le concept de composant est plus récent (milieu des années 90) et apporte la notion de réutilisabilité basée sur la composition (cf. fig. 21) au lieu de l'héritage pour l'objet.
Un composant peut être vu en quelques sortes comme une extension d'un objet et plus particulièrement comme une brique logicielle dont l'intérêt est de pouvoir être assemblé afin d'obtenir un tout.

Cette approche a l'avantage de mettre l'architecture au centre du processus de conception logiciel et de ce fait, de raisonner au niveau d'un système, ce qui permet de s'élever à un niveau d'abstraction supérieur. Cela facilite l'interopérabilité entre les différents langages et les environnements informatiques. Au final, le composant diminue la complexité des applications et plus particulièrement les applications distribuées. Il permet également d'augmenter la productivité et d'améliorer la qualité logicielle.

De façon pratique un composant correspond à un fichier avec une extension : "dll" ou "exe".

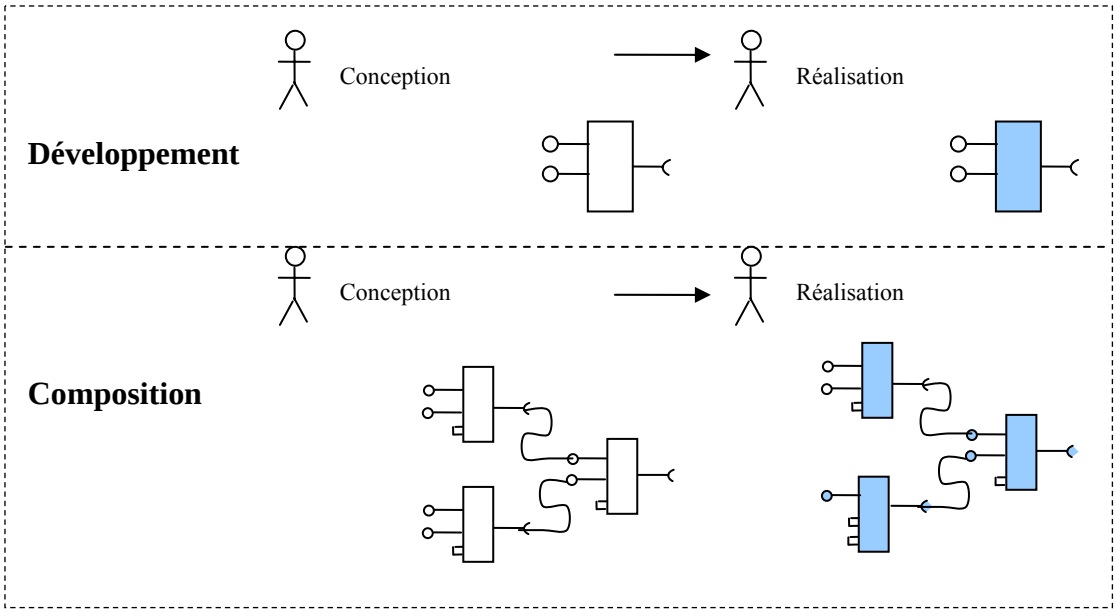

Fig. 21 : Cycle de développement d'un middleware avec composants.

Ainsi, les développeurs manipulent tous le même composant quelque soit l'application. Ils utilisent de ce fait, les mêmes bibliothèques architecturées, ce qui a l'avantage d'écrire moins de lignes de code au final et par conséquent de diminuer la probabilité d'erreurs.

On peut utiliser les fonctionnalités (API) à travers une interface sans forcément avoir accès au code du composant.

Cette nouvelle approche du développement d'applications améliore la qualité de conception, la modularité, la réutilisabilité, la portabilité, l'adéquation aux architectures applicatives réseau. Elle permet en outre de conceptualiser les différents éléments d'un composant et son intégration dans une application.

De façon plus globale, l'intergiciel à composant est utilisé pour concevoir, développer et déployer facilement des applications réparties afin d'assurer l'hétérogénéité, et l'interopérabilité ainsi que la réutilisation de modules logiciels dont le code métier est encapsulé dans des composants et le code système est géré par les conteneurs.

II.3.1.1. Le framework par composants

La définition d'un framework (qu'on pourrait traduire par cadriciel) donnée par Ralph Johnson [Johnson 1997] est "*a framework is a reusable design of a system that is represented by a set of abstract classes and the way their instances interact*"[1], elle peut être complétée par celle donnée par lui aussi en 1993 avec "*a framework is the skeleton of an application that can be customized by an application developpe*" [2] [Johnson 1993].

C'est en effet, un ensemble d'outils, de formalismes, de langages, de bibliothèques de classes qui permettent de construire une application selon un modèle à composants. De ce fait, un framework comporte :

- un formalisme pour décrire l'assemblage de composants,
- un formalisme qui permet de définir la structure interne d'un composant,
- un compilateur pour formalismes.

Ci-après, nous présentons plusieurs modèles de composants sans expliciter forcément le framework qui les accompagnent.

II.3.1.2. Cas COM

Le modèle *COM* (Component Object Model) de Microsoft [Box 1998] a été conçu pour résoudre le problème de l'interopérabilité des composants provenant des différents éditeurs et qui peuvent par conséquent, être écris dans des langages différents même s'il est très lié au langage C++.

Comme tout composant, on peut utiliser COM sans connaitre la réalité physique de son fonctionnement. Le composant COM propose une ou plusieurs interfaces qui correspondent, en vérité à un appel de fonctions d'un autre composant (cf. fig. 22). C'est pour cette raison qu'on parle d'interface de prototype du composant. C'est au niveau de l'interface que portent les règles COM et elles doivent respecter les conditions décrites ci-dessous.

- L'interface doit être identifiée de façon unique. On peut dire qu'une interface est identifiée par un "Interface Identifier" (IDD) qui fait partie en fait d'un "Globally Unique Identifier" (GUID). Elle implique un identifiant unique au niveau de la base de registre d'un poste fonctionnant forcément avec un système d'exploitation Microsoft.

[1] Une tentative de traduction par nous est : "*un framework est un modèle réutilisable de tout ou d'une partie d'un système qui est représenté par un ensemble de classes abstraites et de la façon dont leur instances s'utilisent".*

[2] Tentative de traduction : "*un framework correspond à un squelette d'une application qui peut être personnalisé par un développeur d'applications*"

- L'interface ne doit pas être changée, une fois publiée. La raison est due au fait que l'application ne doit pas être modifiée lorsqu'on modifie le composant. On peut faire évoluer le composant en lui ajoutant de nouvelles interfaces donnant ainsi accès à d'autres fonctions mais on ne peut pas modifier celle déjà existante.

- L'interface doit être dérivée d'une interface racine, nommé IUnknown. L'interface IUnknown définit deux règles qui sont la navigation entre les interfaces et la durée de vie du composant. Ainsi, on peut obtenir une référence sur n'importe quelle interface d'un même objet, du moment où il existe une référence à ces objets.

Le modèle COM n'est pas le plus facile à implémenter même s'il a été novateur à son époque.

Microsoft a introduit une version améliorée de ce modèle avec COM+ (accent mis sur l'aspect distribué) mais c'est plutôt ".Net" qui vise à remplacer COM en adoptant beaucoup de ses idées.

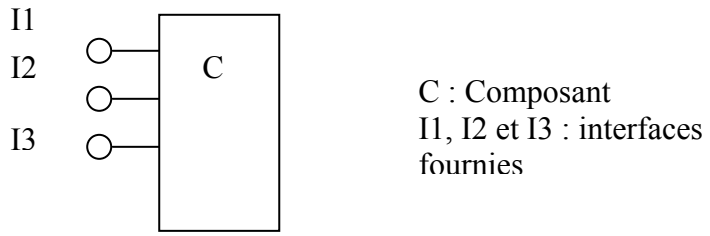

Fig. 22 : Modèle logique d'un composant COM.

II.3.1.3. Cas JavaBeans

Sun a créé ce modèle de composant en 1997 [BEAN 1997] avec l'objectif principal de simplifier la construction d'applications en permettant l'assemblage par l'intermédiaire d'un outil visuel. Ce modèle est écrit en Java et permet de créer des composants suivant un ensemble de conventions de programmation et un ensemble de normes lexicales. A leur construction, ils n'ont aucune référence vers d'autres composants. Ils peuvent être assemblés par des outils externes mais la composition se fait par le framework.

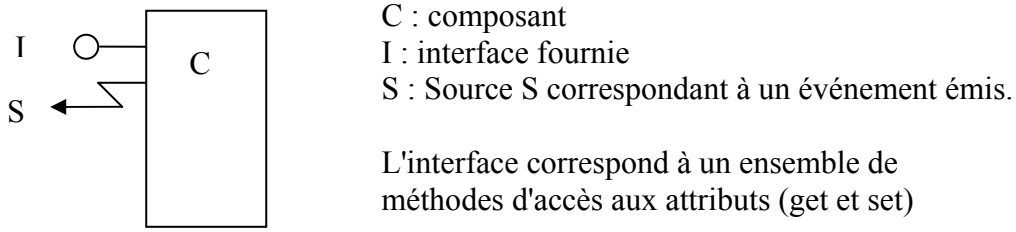

Fig. 23 : Modèle logique d'un composant JavaBeans

II.3.1.4. Cas EJB : Enterprise JavaBeans

Le modèle à composant EJB de Sun [EJB 2001] est plutôt utilisé pour les applications réparties construites selon une architecture en trois tiers. Un tiers se positionnant au niveau du client, un autre au niveau du serveur, et le dernier tiers au niveau de la base de données. Cette architecture nécessite des caractéristiques non fonctionnelles comme la transaction, la sécurité, la distribution et la persistance.

Le modèle EJB supporte mal que les applications du tiers du milieu soient distribuées et est plutôt orienté pour les composants Java situés au niveau du serveur (cf. fig. 24). Ils sont aussi dédiés à l'encapsulation de la logique métier d'une application. On peut utiliser comme container EJB le serveur d'application J2EE développé en open source : JOnAS (Java Open Application Server).

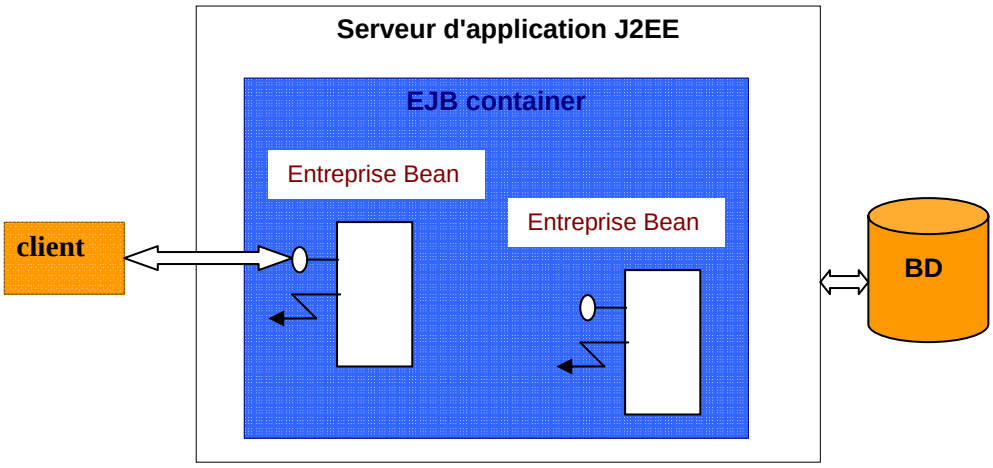

Fig. 24 : Architecture de la technologie EJB.

II.3.1.5. Cas Fractal

Le modèle de composant Fractal [Fractal 2004] a été défini par France Telecom R&D et l'INRIA. Il peut être implémenté dans différents langages comme Java, C, C++, SmallTalk et les langages liés à la plateforme ".NET" comme C# entre autres.

Avec le modèle de composant Fractal, on a la possibilité d'assembler des composants qui appartiennent à plusieurs composites qui ne sont pas imbriqués mais qui se chevauchent. Cette caractéristique est intéressante pour modéliser des ressources partagées. L'architecture d'un système de composant est décrite soit dans le code métier soit dans le fichier de description. On peut assembler des composants de façon dynamique, cela indique qu'ils sont modifiables à l'exécution.
Le modèle est basé sur les principes suivants :

- Composants composites : pour avoir une vision uniforme des applications à différents niveaux d'abstraction.

- Composants partagés pour permettre de modéliser les ressources.
- Capacité d'introspection afin de surveiller l'exécution d'un système.
- Capacité de reconfiguration pour permettre de déployer et de configurer dynamiquement un système.

C'est un modèle qui est bien adapté à la construction des systèmes complexes comme les intergiciels.

II.3.1.6. Cas CCM : CORBA Component Model

Ce modèle [CCM 2006] permet de définir des composants dans une architecture distribuée CORBA qui est compatible avec les structures EJB.

Les applications écrites sous CCM peuvent être distribuées sur plusieurs sites contrairement à celle écrite avec EJB.

Avec CCM, on peut intégrer des clients hétérogènes écrits dans des langages différents. Il est possible d'intégrer des clients hétérogènes écris en C, C++ ou Java, partageant les mêmes contextes. C'est un des modèles les plus complets (cf. fig. 25). Il possède un langage de description externe de composants (OMG IDL3) qui correspond en fait, à une extension du langage IDL (Interface Définition Language) de CORBA 2.0.

L'architecture et l'API proposées par le CORBA Component Model sont sur différents points, très intéressants. En effet, le CCM apporte une solution pour définir un composant tout en considérant différents aspects liés à la notion de composant, aux architectures distribuées, aux différents langages, aux plateformes et également à la gestion des instances des composants, de leur cycle de vie et à la phase de déploiement.

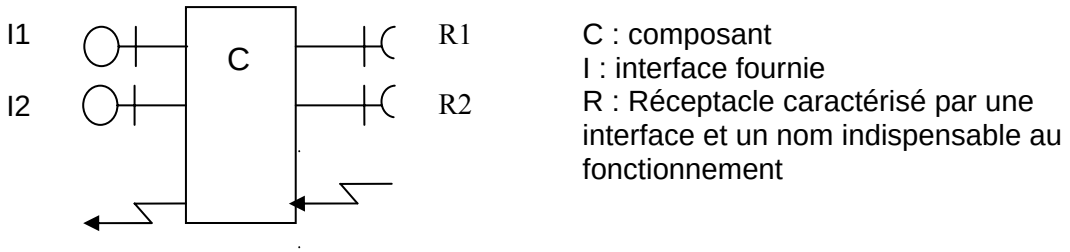

Fig. 25 : Modèle logique d'un composant CCM

II.3.1.7. Conclusion

L'approche par composants est fondée sur un cycle de vie de développement où il existe deux parties distinctes qui sont la création d'un composant et son assemblage. Ces deux parties seront réalisées à des moments et/ou par des acteurs différents. Parmi les

différents modèles étudiés, JavaBean, EJB et CCM sont plutôt spécialisés par rapport au domaine d'application alors que COM et Fractal sont plus génériques. Le modèle JavaBean a aussi l'avantage d'être plus flexible que les modèles EJB, CCM et Fractal grâce au fait qu'il ne contient aucune référence à d'autres composants. Cela permet de dire qu'on n'est pas obligé de compiler le code du composant avant la création de la liaison avec un autre composant.
Le modèle à composants a le mérite de proposer des mécanismes de composition avec en plus, des modèles de développement simples, il permet de développer des middlewares par l'ajout, le retrait ou le remplacement de composants.

II.3.2. Modèle et plateformes à base de services

Les modèles et plateformes à base de services sont basés sur l'architecture orienté service (SAO, *Service Oriented Architecture*) [Bieber] qui définit le service comme brique de base pour la construction d'application.

Le service est décrit comme une fonctionnalité dont la caractéristique est d'être contractuelle. De façon générale, on peut définir un service comme un comportement défini par contrat, qui est implémenté et fourni par un fournisseur pour être ensuite utilisé par un demandeur, sur la base exclusive de ce contrat. Ce service a l'avantage d'être utilisé dans différentes applications quelques soit le nombre de sollicitation. Il peut être accessible à travers une ou plusieurs interfaces. L'interface a un rôle de description de l'interaction entre le client et le fournisseur du service. Elle définit en effet, les opérations et les structures de données qui participent à la réalisation du service.

La découverte de services se fait par l'interaction de trois acteurs : le fournisseur de services, le demandeur de services et le registre de services dont nous détaillons ci-dessous les rôles respectifs :

- Le fournisseur de services a pour rôle de produire des services qui ont une disponibilité dynamique.
- Le demandeur de services correspond au client.
- Le registre de services contient un ensemble de descripteurs de services ainsi que des références vers les fournisseurs. Il se situe en quelque sorte entre le demandeur et le fournisseur de service (cf. fig. 26).

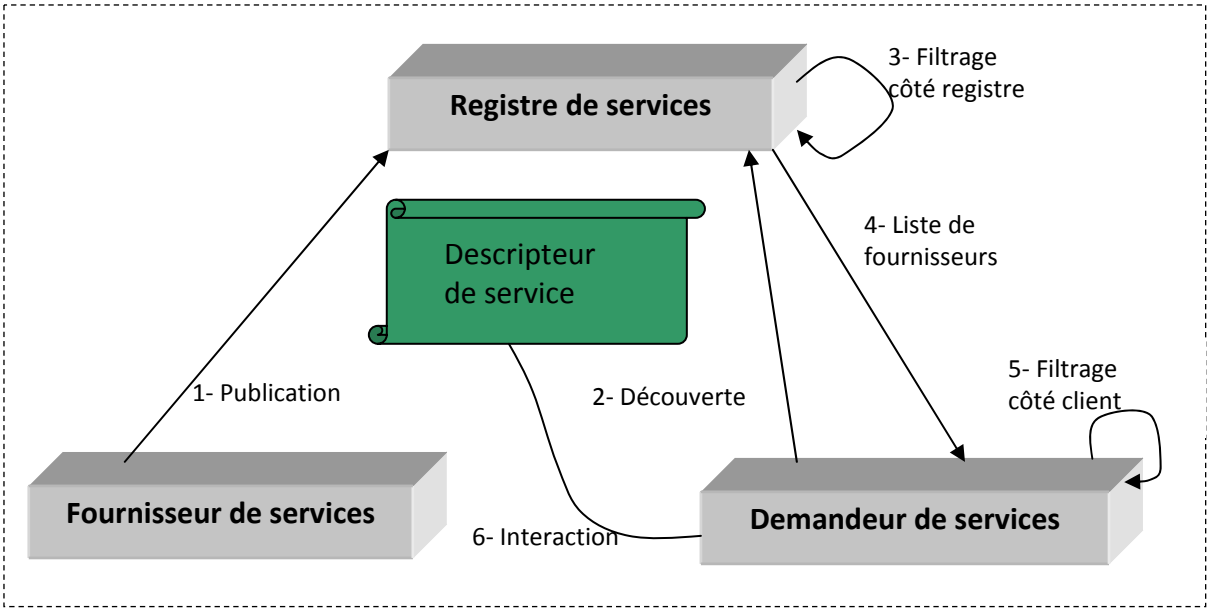

Fig. 26 : Modèle orienté service

Comme avec le modèle à composants, l'architecture orientée service permet la réutilisabilité des services, l'abstraction en limitant dans ce cas, le service au niveau du contrat et la composition pour former des services composites.

En plus, cette architecture apporte des notions nouvelles qui ont d'ailleurs considérablement influencé le domaine du génie logiciel et ont été adoptées par la réalité mixte ou l'informatique ubiquitaire. Ces notions sont :

- le contrat de service qui formalise la communication entre les services en les obligeant à y adhérer et en leur offrant de ce fait, une description de ce qui est offert ou attendu.
- L'indépendance des services des uns par rapport aux autres et assurant ainsi leur autonomie.
- L'encapsulation des services, permettant à n'importe quel programme d'une application d'être exécuté.
- La dynamicité plus grande en minimisant les dépendances entre les services et assurant ainsi un couplage faible.
- La découverte de service qui se fait au moment même où il s'exécute, de façon dynamique pour ainsi dire.

Ci-après, nous présentons les produits basés sur ce paradigme.

II.3.2.1. Courtier CORBA

CORBA (Common Object Request Broker Architecture) [CORBA 2008] a été initié par plusieurs sociétés comme Sun (avant son rachat par Oracle), IBM, Oracle au sein de l'Object Management Group. CORBA est une architecture logicielle qui permet de développer des composants et des ORB (*Object Request Broker*) (qu'on pourrait traduire

par *courtier de requêtes à objet*) afin de faciliter l'interaction des objets distribués. Un ORB correspond en effet, à un ensemble de classes qui implémente un canal de communication par lequel les objets envoient des requêtes et reçoivent des réponses de manière transparente. Il s'agit en vérité de services qui correspondent à des méthodes d'autres objets distribués. La plupart des ORBs s'appuient sur la norme CORBA (hormis la technologie COM de Microsoft). CORBA permet de rendre les objets indépendants du langage avec lequel ils ont été écrits. Ainsi, un objet écrit en Java peut communiquer avec un autre écrit en C++. L'objet CORBA n'implémente aucune méthode et correspond à une interface écrite avec le langage IDL (*Interface Définition Language*). Ce langage permet de décrire les services (appelés *type de service*) qui sont rajoutés dans l'annuaire de spécification de service. Ainsi, il faut récupérer une référence au niveau de l'annuaire de spécification de service (via l'ORB) et publier le type de service. Un fournisseur publie son *offre de services* à la condition que dans un premier temps le type de service ait été publié. Le retrait de service se fait également en deux étapes. Les fournisseurs retirent d'abord le service grâce à l'utilisation de la fonction "withdraw" de l'annuaire de service. Dans un second temps, c'est l'administrateur qui décide de supprimer le type de service par l'intermédiaire de la méthode "remove_type" au niveau de l'annuaire de spécifications de services.

Un consommateur de service se connecte à l'annuaire de service afin de rechercher un service. Sa requête spécifie le type de service. Il reçoit en retour une liste d'offres qui lui permet d'accéder au service à travers l'ORB. Il peut ainsi utiliser le service en invoquant les méthodes décrites dans l'interface du service. Cette demande se fait d'une façon générale à travers *IIOP* (Internet Inter-ORB Protocol) (cf. fig. 27) car on peut aussi utiliser Java RMI.

CORBA ne sait pas fournir par défaut un service à partir d'autres services. La composition de service doit en effet, être codée complètement afin d'assurer le dynamisme manuellement.

On peut dire, en guise de conclusion que CORBA sait proposer un environnement de développement réparti suivant l'approche à service, mais il ne supporte pas la recherche passive qui empêche malheureusement de réaliser des middlewares réellement dynamiques.

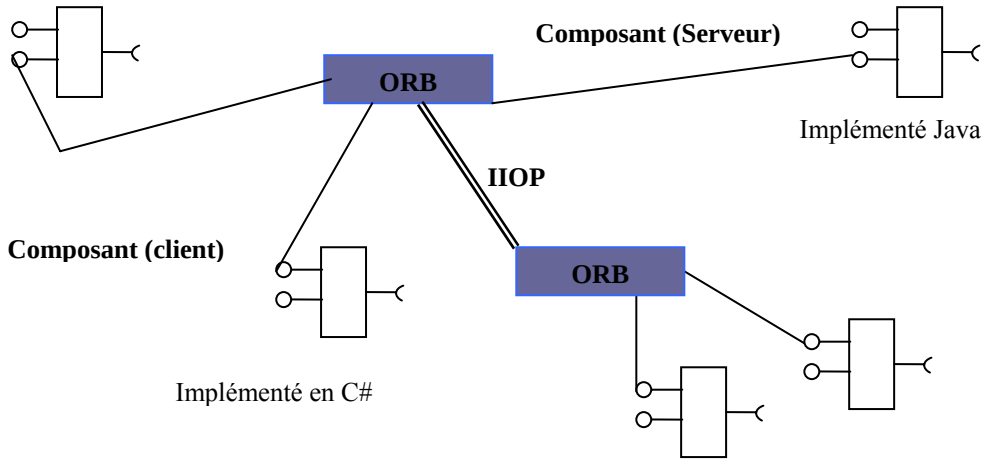

Fig. 27 : Déploiement d'applications (ou middleware) avec CORBA

II.3.2.2. Service Web

L'architecture à service web [Service Web 2004] (cf. fig. 28) aide à l'interaction d'applications hétérogènes par l'intermédiaire de services offerts. Les interfaces et les attaches publiques sont définies et décrits grâce à une description en XML. On peut utiliser plusieurs Services Web afin de créer un nouveau service.

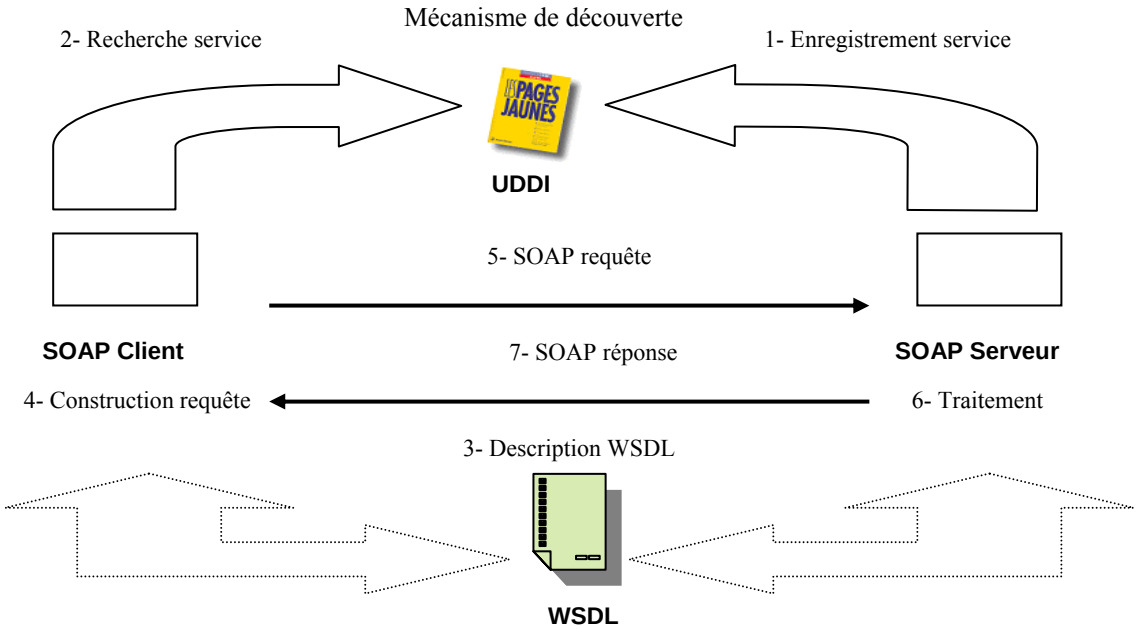

Fig. 28 : Architecture à base des services Web

Les trois éléments de base du Service Web (cf. fig. 29) utilisent des protocoles et des langages ouverts qui assurent l'interopérabilité entre les services et les systèmes.

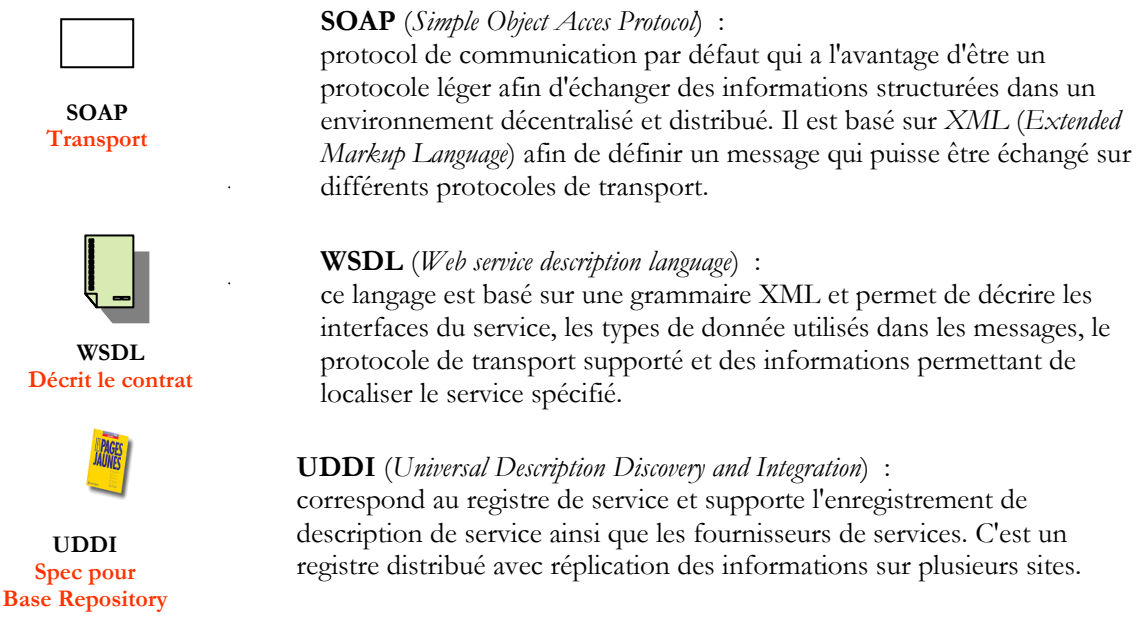

Fig. 29 : Eléments de base du Service Web

II.3.2.3. Jini

La technologie Jini [Jini River] repose sur une architecture orientée service qui partage beaucoup de ces concepts avec l'architecture de courtier CORBA. A l'origine, l'éditeur était la société Sun Microsystems qui l'a transféré depuis à la fondation Apache sous le nom de projet River.

Jini repose sur *Java* et *Java RMI* (Remote Method Invocation) et définit le concept de *fédération*. On peut dire qu'une fédération est un groupement de ressources (aussi bien matérielle que logicielle) reliées dans un même réseau. Chaque élément d'une fédération peut fournir ou utiliser un service. Jini offre un service de recherche (*lookup service*) qui joue le rôle de courtier.

Le principe repose sur 3 acteurs selon le mécanisme suivant : le fournisseur de service (*service provider*) recherche un service d'enregistrement (*lookup Service*). Ensuite le fournisseur de service enregistre l'objet service ainsi que ses attributs dans le service d'enregistrement. Le client vient à demander un service selon ses attributs et une copie lui est envoyée. Le client interagit directement avec le fournisseur du service par l'intermédiaire de l'objet service. Jini supporte la libération des objets par expiration de *bail* contrairement au courtier CORBA où la communication se fait par un appel à distance entre le demandeur et l'objet. Cette notion de bail caractérise la technologie Jini et est utilisée lorsque le fournisseur veut publier son service. Il doit renouveler le bail lorsqu'il expire sous peine de voir l'offre de service supprimée du service de recherche.

Une autre caractéristique de Jini est que client et fournisseur doivent trouver un registre au préalable avant d'interagir. Un registre est accessible par une adresse IP fixe ou découverte à partir d'une demande diffusée. Jini supporte la découverte passive. Le consommateur doit s'enregistrer auprès du service de recherche. Il informe du type de service souhaité et est par la suite informé des arrivées et départs du fournisseur l'intéressant.

Jini (tout comme CORBA d'ailleurs) ne propose pas de modèle de composition, et c'est le développeur qui doit composer les services et gérer l'aspect de dynamiciter des compositions.

En conclusion, on peut dire que Jini propose toute l'infrastructure pour créer des middlewares dynamiques. Cette technologie sait en effet, gérer la recherche active et passive et de ce fait, la notion de dynamisme est donc gérable. Cependant, il existe des défauts dans l'implémentation de référence qui l'ont empêché de rencontrer le succès espéré.

II.3.2.4. OSGi

Le modèle OSGi (*Open Service Gateway Initiative*) [OSGi 2009] a été proposé par un consortium industriel en 1999. Il devait répondre aux critères de spécifications ouvertes afin de développer et déployer des services administrés dans des réseaux résidentiels comme des modems ADSL ou des décodeurs de TV numérique. Depuis, le champ d'application de la plateforme OSGi s'est élargi au domaine de l'électronique grand public, de l'industrie, de l'automobile, des télécommunications mobiles, de la domotique.

Au niveau de la plateforme OSGi, on a des unités logiques et physique qui déploient et livrent les services. Ces unités correspondent à des *bundles* et se matérialisent par des fichiers JAR qui contiennent le code binaire des classes, les fichiers de configuration ou des images ainsi que des bibliothèques de code natives (i.e. dépendant du processeur et du système d'exploitation).

Il est possible d'installer, supprimer, redémarrer des bundles sans interrompre la plateforme OSGi. Chaque bundle peut fournir et demander des services d'autres bundles. Sachant qu'OSGi permet la gestion dynamique des unités de déploiement, les fournisseurs de service peuvent apparaitre et disparaitre à l'exécution. Cette architecture propose aussi un *annuaire de service* (*service registry*). Le fournisseur de service a accès à l'annuaire via le *bundle context* qui correspond à un objet spécifique injecté à l'activation du bundle. Pour s'enregistrer, le fournisseur spécifie, le service choisi et les propriétés relatives au fournisseur. A la fin de son enregistrement, il reçoit un talon d'enregistrement (*service registration*) lui permettant de retirer le service.

Le consommateur accède à l'annuaire en utilisant également le bundle context. Pour rechercher un service, il interroge l'annuaire à travers un filtre qui utilise le langage LDAP. Lorsque le consommateur cesse d'utiliser le service, il doit relâcher le service et libérer les références sur l'objet de service. Ceci évite le blocage des opérations sur le bundle lors de désinstallation, de mise à jour ou de réinstallation en libérant de la mémoire, et également les classes susceptibles d'être contenues dans les nouveaux bundles.

La composition entre les instances de composants se fait de manière dynamique suivant la disponibilité des ressources.

OSGi permet la recherche passive et ne définit pas clairement le modèle de composition même si la notion de dynamisme est assurée. La composition avec le dynamisme des services n'est pas triviale car elle peu faire échouer une composition. En effet, le développeur doit gérer l'enregistrement et le retrait du service de sa composition ainsi que le traitement de vérification du relâchement des services. Le développement d'application est aussi très délicat en raison de la dépendance entre les bundles et la dépendance des services.

On peut conclure en disant qu'OSGi fournit toutes les primitives lié à l'approche à service nécessaire à la création d'application dynamique. Il fournit également des outils pour l'administration et le déploiement. Mais le développement avec cette plateforme n'est pas aisé à cause des modèles fournis.

Le modèle à composants OSGi peut être considéré comme un modèle à composant qui fait référence à un groupe de services lié par des caractéristiques physiques (ils partagent le même fichier de déploiement).

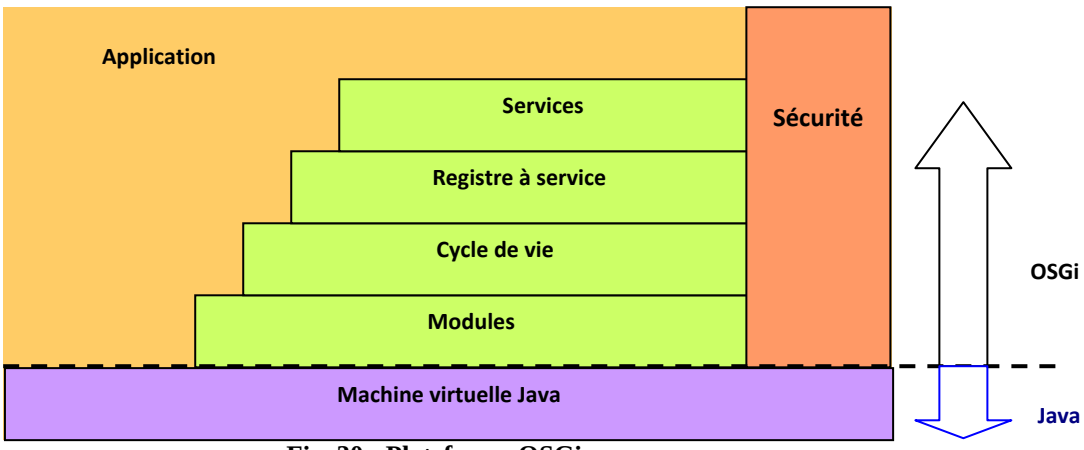

Fig. 30 : Plateforme OSGi

II.3.2.5. UPnP

UPnP (Universal Plug and Play) [UPnP 2010] est un protocole réseau spécifié par UPnP Forum et destiné à la mise en place de réseaux domotique. L'architecture est basée sur

les protocoles standards utilisés avec internet afin d'assurer une architecture à service dynamique. On retrouve ainsi les protocoles TCP ou UDP de la couche transport du modèle OSI, IP pour la couche réseau, HTTP pour la couche session et XML pour la couche présentation. C'est une architecture ouverte et distribuée qui permet une certaine indépendance par rapport au support, et au système d'exploitation ainsi que du langage de programmation.

A chaque connexion d'un appareil, UPnP se met en activité suivant les étapes de fonctionnement décrit ci-dessous :

- **Adressage** : A la connexion d'un appareil, UPnP permet de joindre dynamiquement le réseau et d'obtenir une adresse IP. S'il n'y a pas de serveurs DHCP, l'appareil s'assignera lui même une adresse grâce à la technique Zeroconf (allocation dynamique d'adresse due au standard IPv4 et IPv6).

- **Découverte** : Une fois l'adresse obtenue, l'appareil détecte les autres appareils par l'intermédiaire du protocole de découverte : SSDP (Simple Service Discovery Protocol).

- **Description** : Cette étape est réalisée par le point de contrôle avec la récupération du fichier de description de l'appareil (nom modèle, numéro de série nom fournisseur). Ce fichier est au format XML.

- **Contrôle** : Le contrôle consiste à l'envoie d'ordre au service du dispositif. Les messages de contrôles sont décrits en XML et utilisent SOAP.

- **Notification d'événement :** UPnP propose la possibilité d'être notifié des changements d'états d'un dispositif. Lorsqu'une variable d'état de type *evented* est modifiée, cela émet une notification aux points de contrôle qui se sont abonnés pour les recevoir. Lors du changement des variables, le service publie des mises à jour qui sont au format XML de type GENA (General Event Notification Architecture).

- **Présentation** : La dernière étape correspond à la prise de contrôle d'un dispositif ou simplement à la vérification de son état par l'intermédiaire d'un butineur (ou browser). Pour ceci, l'appareil doit avoir un serveur http embarqué ainsi qu'une page de présentation web de configuration du dispositif.

II.3.2.6. Service Web Pour dispositif : DPWS

Les services web pour dispositif : *DPWS* (Device Profile for Web Service) [DPWS, OASIS 2009] est le nom donné à la révision 2 des spécifications UPnP. L'intérêt de cette technologie est l'adoption de standards concernant les services web en incluant de nombreuses extensions basées sur langage WSDL et le protocole SOAP. Ces extensions sont (cf. fig. 31) :

- WS-Security fournit les mécanismes pour assurer l'intégrité et la confidentialité des données grâce à l'encryptage, la signature.

- WS-Policy définit la qualité de service (QoS) nécessaire à la communication de services.
- WS-Addressing permet de fournir un mécanisme d'adressage des services web.
- WS-Discovery. Lors de la première étape de découverte de service, DPWS fait appel à ce service en utilisant un annuaire local de service et en utilisant également une syntaxe LDAP qui supporte des filtres avec plusieurs critères de recherche.
- WS-Eventing définit la façon de souscrire et d'envoyer un événement.
- WS-MetadataExchange concerne la définition du format de fichier de description des services.

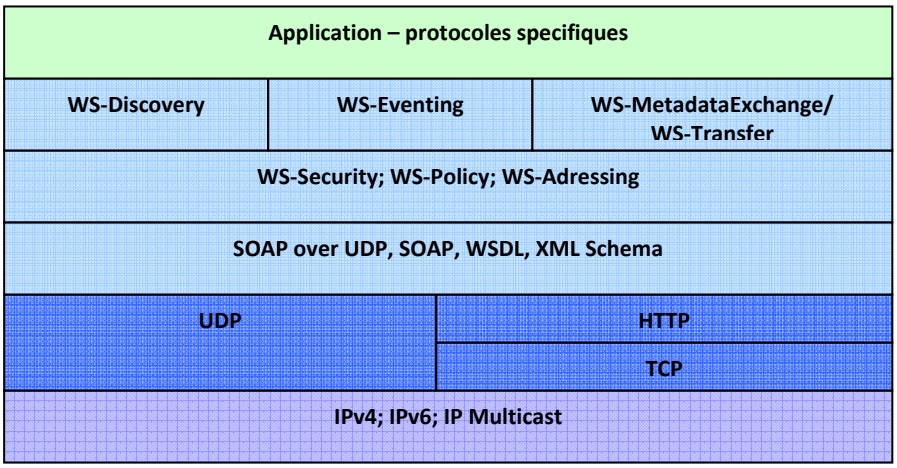

Fig. 31 : Pile des protocoles DPWS.

II.3.2.7. Conclusion

L'architecture à service répond, contrairement à l'architecture par composant, au besoin de dynamisme propre à la réalité mixte (et l'informatique ubiquitaire). La mise en place d'application à service dynamique est toutefois plus complexe à programmer. Il n'est pas également possible de spécifier la dépendance de services et encore moins de les composer. Enfin, la notion d'implémentation de service n'est pas supportée.

II.3.3. Modèle et plateformes à base de composants orientés services

Les principes du paradigme composant orienté services ont été introduits par H. Cervantes et R. S. Hall en 2004 [Cervantes 2004] et a été depuis, normalisé par le consortium OASIS [OASIS, SCA 2007]. Le but de ce regroupement est justement de bénéficier de l'avantage du modèle à composants qui comme nous l'avons vu, apporte la simplification du modèle de développement et la description de composition, ainsi que de l'avantage du modèle à service concernant le faible couplage et le dynamisme (cf. fig.32).

Au final, on obtient une simplification de conception des applications orienté service, tout en gardant les avantages propres à ce modèle.
En effet, cette approche permet la création de fournisseurs de service par l'approche à composant. Aussi la plate-forme permet de gérer les liaisons dynamiques propres à l'approche à service dans une partie non fonctionnelle d'un composant.

On peut résumer les principes de ce modèle par les points ci-dessous :

- Le composant doit implémenter une spécification de service.
- Le service correspond à une fonctionnalité fournie.
- Le service est décrit par une spécification.
- Une composition est décrite comme une spécification de service.

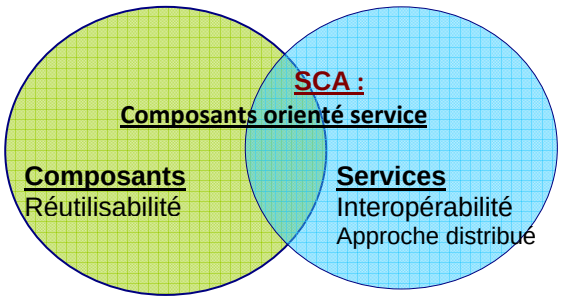

Fig. 32 : Modèle à composants orienté services.

Ci-après, nous présentons deux implémentations de cette approche : SCA et iPOJO.

II.3.3.1. Modèle SCA

Le modèle SCA (Service Component Architecture) a été créé par un consortium d'industriel regroupé au sein d'OSOA (Open Service Oriented Architecture) [SCA, OSOA 2010]. Ce modèle a l'avantage de pouvoir créer des composants à service dans différents langages qui ont la possibilité d'être exécuter sur différentes machines. Pour le dire de façon plus précise, SCA se veut indépendante d'un langage de programmation et du protocole d'appel de services (services web, RMI, CORBA…) pour se consacrer davantage à l'écriture de la partie métier.

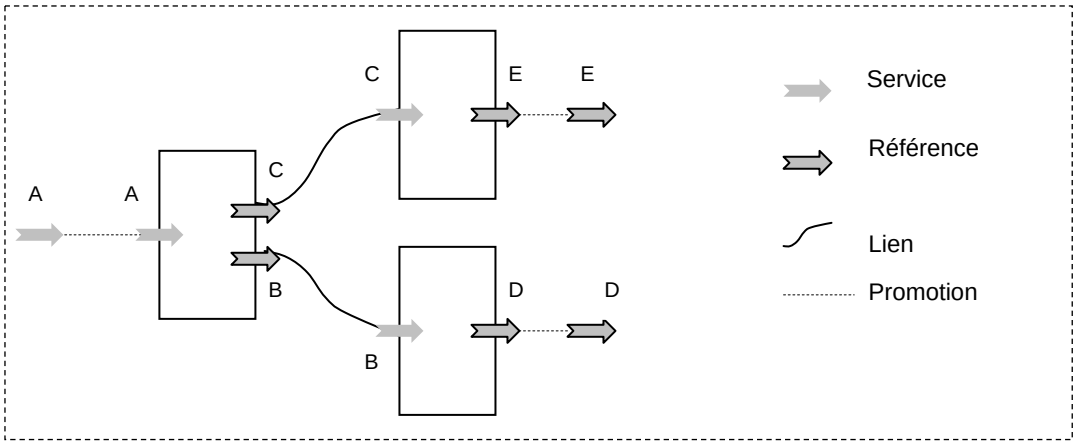

Fig. 33 : Exemple d'application SCA

II.3.3.2. iPOJO

iPOJO (*injected Plain Old Java Object*) [iPOJO] est hébergée par le projet open source : Apache Felix, bien qu'il ait été conçu à l'origine par l'équipe ADELE du laboratoire LIG (Laboratoire d'informatique de Grenoble). Apache Felix est implémenté avec OSGi et l'insertion d'iPOJO apporte la notion de composant orienté service. De ce fait, la plateforme iPOJO respecte les caractéristiques d'OSGi et adopte le langage Java. Cette plateforme est également centralisée et assure un haut degré de dynamisme.

Les caractéristiques propres à iPOJO sont :

- création simple de composants utilisant des services.
- langage de définition d'applications à services.

Dans le modèle de base, iPOJO utilise une notion provenant de l'approche composant qui est le système de conteneur. Le conteneur gère le cycle de vie du composant ainsi que les communications avec l'extérieur. Un conteneur est formé d'un ensemble de gestionnaires appelés "handler". Chaque handler a pour rôle de gérer la partie non métier du composant, appelé communément partie non fonctionnel. Par exemple, on peut citer l'aspect gestion du dynamisme vers les services. L'utilisateur de la plateforme d'iPOJO peut rajouter des handlers pour son besoin spécifique et qui ne sont pas forcément présent dans le conteneur. Il peut ainsi spécialiser la plateforme pour un domaine métier particulier.

Les handlers importants, fournis de base sont :

- le handler qui permet la publication de services implémentés de façon automatique par le composant. En effet, la publication et le retrait se font en fonction de la capacité même du composant à pouvoir le faire.
- Le handler qui gère la dépendance de service. Il peut en effet, injecter les services requis, publier les services fournis et découvrir les services utiles. Le cycle de vie du composant en dépend. Ce qui se traduit par le fait que le composant reste en fonctionnement ou bascule dans l'état invalide.
- Le handler qui gère le démarrage, l'exécution, l'arrêt (cycle de vie) des handlers.

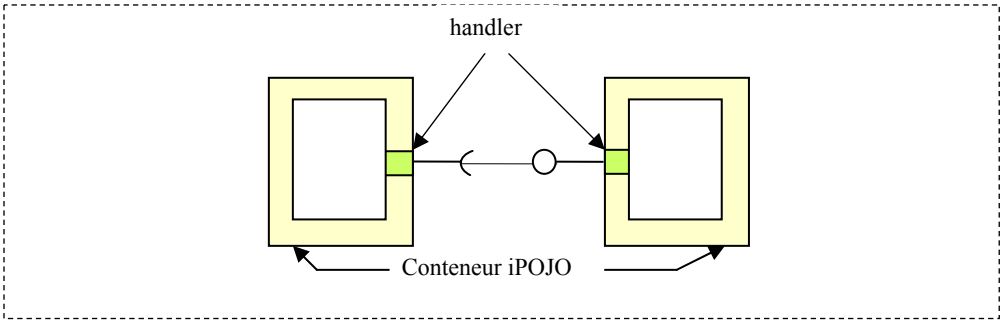

Fig. 34 : Conteneur iPOJO

L'assemblage pour la création d'application se fait non pas sur des composants mais sur un ensemble de spécification de services. iPOJO introduit la notion de composite qui correspond à l'assemblage de composant à l'exécution de l'application. Lors de l'assemblage, on peut composer les services qu'on met dans un composite.

II.3.3.3. Conclusion

Le modèle à composants orienté service à l'avantage de récupérer les qualités des modèles à composant et à service. Ce qui se traduit par une simplification de développement de middleware et apporte les avantages suivants :

- Un composant peut implémenter des spécifications de service.
- Les spécifications de service correspondent à des compositions.
- Un service correspond à une fonctionnalité fournie.
- Un service peut être spécifié suivant des informations comportementales, sémantiques, syntaxiques.

II.3.4. Etat de l'art d'intergiciels

Il existe plusieurs intergiciels qui ont adopté un ou plusieurs des paradigmes évoqués précédemment. Dans ce sous-chapitre nous présentons un état de l'art afin de vérifier qu'il n'en existe pas déjà un qui puisse répondre complètement ou en partie à notre besoin. En fonction de ce constat, nous présentons notre choix concernant le middleware et la plateforme pouvant supporter notre problématique.

II.3.4.1. AMIGO

Amigo (AMbient Intelligence to GO) [AMIGO 2008] est un intergiciel open-source orienté service. Il est le fruit d'un projet de recherche porté par 16 partenaires issues de sociétés commerciales, d'instituts de recherche et d'universités qui a pris fin en 2007.

Le noyau correspond à l'architecture pour service web avec un développement conséquent de services supplémentaires. Ce middleware peut se décomposer en trois parties :

- la partie : *"Base middleware"* apporte les fonctionnalités nécessaires à l'environnement de communication. Cela concerne les moyens de communication et la découverte de services et de périphériques comme Web Service, SLP (Service location protocol) [SLP 1997], UPnP. L'aspect sécurité comme l'autorisation, l'authentification et le chiffrement sont gérés à ce niveau.

- la partie : *"Intelligent User Services"* apporte les fonctionnalités nécessaires au contexte, traite les différents sources d'informations pour établir des prédictions basés sur des modèles. Ainsi l'information est adaptée suivant la situation de l'utilisateur et du changement de son contexte. Voici les composants avec leurs fonctionnalités :
  - *Context management service* (CMS) : c'est à ce niveau que la gestion du contexte se fait, par la prise en compte des informations provenant de sources diverses. Comme exemples, on peut citer : des capteurs, l'activité des utilisateurs ou une application Internet. On peut l'utiliser par l'implémentation de l'API : ContexteSource afin de fournir les informations contextuelles à d'autres composants.
  - *Awareness and notification* : ce service ANS fournit les observations du contexte et envoit des événements aux applications lors d'un changement. Les applications doivent définir des règles qui précisent les changements à notifier. Les utilisateurs peuvent également préciser leurs préférences afin de sélectionner la remonté d'informations approprié du contexte.
  - *Privacy and security* : ce composant permet d'encapsuler la communication et les primitives de cryptographies qui sont utilisées lors de la phase d'authentification.
  - *User modelling and profile management* : permet d'améliorer l'efficacité des services et la convivialité des interfaces afin de présenter des informations pertinentes suivant des modèles et des profils utilisateurs qui correspondent en fait, aux préférences utilisateurs.

- La partie : "*Programming and Deployment Framework*". Cette plateforme permet de développer des services Amigo-Aware en se basant sur "OSGi" et ".Net" et avec l'utilisation de plusieurs protocoles concernant l'aspect communication et découverte. Ainsi les développeurs sont soulagé pour cet aspect spécifique de gestion des protocoles et de découverte, ce qui au final, leur fait gagné un temps non négligeable.

II.3.4.2. CAMidO

L'intergiciel CAMidO (Context Aware MIDdleware based on Ontology meta-model) [CAMIDO 2006] a été créé au sein de l'Institut National des Télécommunication Sud Paris et de l'université d'Evry. Il se base sur le modèle composant en permettant de créer des applications sensibles au contexte. Il a été implémenté au dessus de CCM (cf. CORBA Component Model, § II.3.1.6) et offre un méta-modèle au développeur tout en lui laissant le soin de traiter les aspects de programmation des interfaces fonctionnelles des composants ainsi que la programmation des méthodes propres à la gestion du contexte.

L'approche de CAMidO a été la séparation de la gestion du contexte de celle de l'adaptation. Au niveau de l'architecture, on le retrouve ainsi avec le gestionnaire de contexte (ContextManager) et le gestionnaire d'adaptation dont nous détaillons ci-dessous les fonctionnalités :

- Le "*ContextManager*" a pour rôle d'interagir avec les dispositifs externes pour recueillir les données du contexte afin de les interpréter. Il repose sur les 5 entités suivantes :
    - "*CollectionManager*" supervise la partie capteurs logiciels en charge de récupérer les informations provenant des capteurs physiques.
    - "*ContextAnalyser*" a pour fonction de détecter les changements du contexte.
    - "*ContextInterpreter*" se charge d'interpréter les informations du contexte suivant les règles décrites dans le méta-modèle du middleware.
    - "*InferenceComponent*" apporte des informations au "ContextInterpreter" ainsi qu'au "ContextAnalyser".
    - "*ContextRepository"* conserve les informations relatives au contexte.

- Le gestionnaire d'adaptation se base sur le composant afin d'exploiter l'adaptation. En effet, ce sont les contrôleurs d'adaptation qui jouent précisément ce rôle et sont justement installés au niveau du conteneur d'un composant.

II.3.4.3. CARISMA

CARISMA (Context Aware Reflexive mIddleware System for Mobile Application) [CARISMA 2003] est un intergiciel réflexif[3] qui permet de traiter les changements de l'environnement.

On peut ainsi utiliser XML pour configurer des situations, ce qui permet à l'intergiciel d'adapter les services à l'application. Il existe des API qui permettent de changer le profil

[3] En génie logiciel, ***la réflexvité*** correspond à la capacité qu'à un programme à s'auto modifier suite à un examen de son propre état.

de l'application lors de son exécution. L'intergiciel est vu en effet, comme un fournisseur de services configurable de façon dynamique.

Il peut exister des conflits entre plusieurs applications lors de leur exécution dans une seule instance de l'intergiciel. En effet, on ne peut pas configurer les détections à priori, car les utilisateurs peuvent changer la configuration de l'application lors de son exécution. CARISMA a donc prévu de résoudre ces cas de façon dynamique par une approche producteur, consommateur. Cet intergiciel est composé de 4 composants (cf. fig. 35) :

- *Context Manager* gère la collecte d'information en provenance des dispositifs de l'environnement.
- *Core* fournit les fonctionnalités essentielles comme la gestion de la communication et la découverte de service.
- *Core services* gère les informations relatives du contexte.
- *Application model* permet de créer des applications.

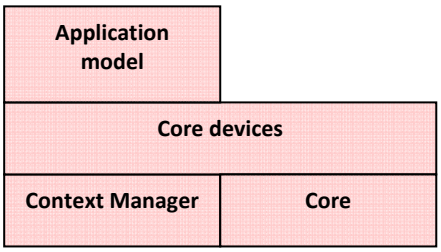

Fig. 35 : Architecture de CARISMA

Cet intergiciel à l'avantage d'être spécialisé dans le traitement de capteurs mais aucun mécanisme d'interprétation n'est prévu.

II.3.4.4. CARMEN

L'intergiciel CARMEN (Context-Aware Ressource Management Environment) a été développé au sein du département DEIS de l'université de Bologne [CARMEN]. Il est basé sur le modèle à composant avec utilisation des services web et de la programmation événementielle pour le traitement de la communication sans fil (Bluetooth et Wifi).

C'est précisément un middleware de gestion des ressources sensibles au contexte, capable de supporter la reconfiguration des services internet sans fil. Il détermine pour cela, le contexte suivant des métadonnées. A cette fin, il utilise la technologie d'agent mobile en tant que proxy de services web.

Son architecture se compose de 2 couches principales (cf. fig.36) : La couche haute, formé par deux composants : "*Context Manager*" et "*Metadata Manager*" et la couche basse formé uniquement par le composant "*Event Manager*" dont nous décrivons ci-dessous les fonctionnalités :

- *Context Manager* a pour rôle d'assurer l'aspect dynamique du contexte.

- *Metadata Manager* gère les différents types des métadonnées du middleware comme les politiques de gestion, les préférences utilisateur, les caractéristiques des ressources. Il est en charge de la mise à jour, l'accès de control dynamique et la gestion de la mobilité.
- *Event Manager* permet de délivrer des événements suivant les changements du contexte et de les diriger soit vers le "*Metadata manager*" pour être utilisé avec les règles appropriées, soit vers le "*Context Manager*" pour en informer le client.

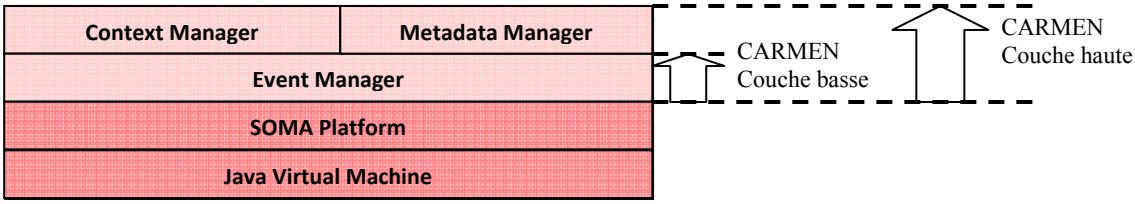

Fig. 36 : Architecture de CARMEN

II.3.4.5. CORTEX

L'intergiciel CORTEX (CO-opérenting Real-time senTient objects : architecture and EXperimental evaluation) [CORTEX 2003] est implémenté avec OpenCOM qui est lui même basé sur la technologie COM (cf. § II.3.1.2). Ce qui a l'avantage de rendre la plateforme efficace, légère et réflexif.

L'architecture de ce middleware est composée suivant 4 structures de composants :

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

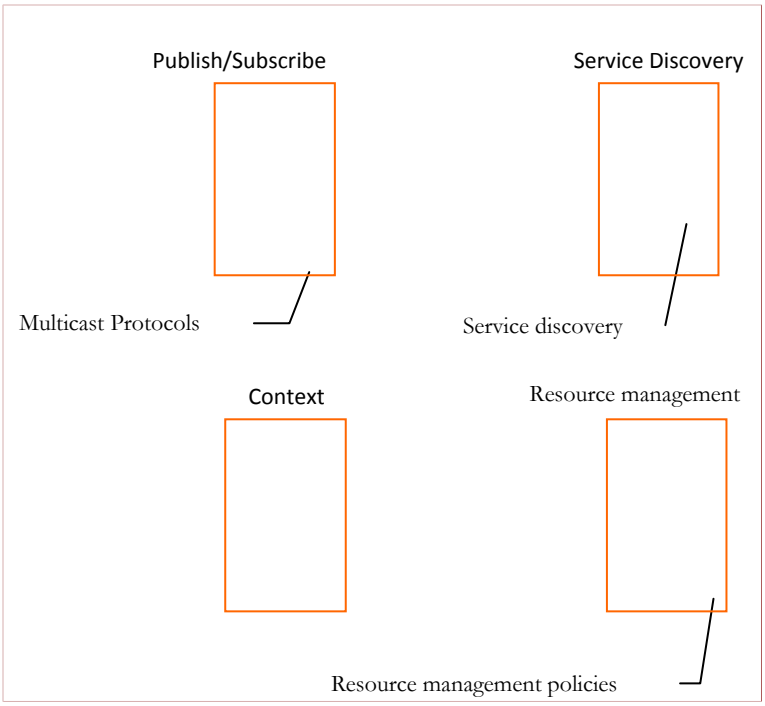

Fig. 37 : Architecture de CORTEX

Le fait que CORTEX utilise un modèle à événement le rend très approprié pour un environnement ad hoc dans lequel les déconnexions sont imprévisibles.

II.3.4.6. GAIA

GAIA [GAIA 2002] est un intergiciel développé par l'université de l'Illinois pour la création d'infrastructures "active spaces", c'est-à-dire d'environnements ubiquitaires. Il gère en effet, les ressources liées à l'environnement ubiquitaire et donne l'emplacement, le contexte et les événements. Il est basé sur GaiaOS, qui peut fonctionner au-dessus de certains systèmes d'exploitation comme Windows2000, WindowsCE et Solaris.

Il utilise TAO [GAIA, TAO 1999] qui est basé sur le courtier CORBA pour prendre en charge la configuration d'exécution. Les applications développées avec GAIA sont aussi basées sur le model CORBA qui correspond à des composants distribués dont la gestion se fait à distance. L'utilisation d'événement lui apporte des qualités supplémentaires de dynamicité de l'environnement et aussi de variabilité au contexte.

II.3.4.7. Oxygen

Le projet Oxygen [Oxygen] a été développé au MIT (Massachusettes Institute of Technology) avec comme intention première de mettre l'homme au centre de la

communication avec son environnent. Cela implique une maitrise de nombreux paramètres comme la communication dans un environnement pervasif, les problèmes liés aux déplacements et leurs conséquences comme l'adaptation et la flexibilité.

L'interface utilisateur propose une communication par la parole et par la vision. Le système se base sur 3 domaines distincts :

- Des capteurs pour l'environnement, appelé "E21s". Ils font office d'interface pour caméra, microphone, écran et d'une façon générale, pour tout dispositif lié à la voix et à la vue. Ils créent ainsi dans une maison ou dans une voiture par exemple, des espaces intelligents.
- Des dispositifs transportés, appelés "H21s" qui gèrent la communication en assurant des points d'accès pour les utilisateurs mobiles. Ils peuvent se reconfigurer pour fonctionner avec plusieurs protocoles et accomplir ainsi des fonctions utiles pour servir les téléphones portables, les radios, les assistants personnels.
- Le réseau auto-configurable appelé "N21s" qui permet de localiser les machines, les personnes, les services qui sont à solliciter.

Le fonctionnement commence par la modélisation d'une personne. Ensuite le réseau se comporte de façon dynamique en fonction de la position des dispositifs. Pour cela, on distingue quatre types de dispositifs : dispositif fixe, mobile, dispositif capteur/actionneur et dispositif avec logiciel. Enfin, les utilisateurs obtiennent des ressources fournies par l'environnement en fonction de la définition des règles de connexion.

II.3.4.8. RCSM

RCSM (Reconfigurable and Context-Sensitive Middleware) [RCSM 2004] est un intergiciel conçu par l'université de l'Arizona, selon le modèle à service du courtier ORB (cf. § II.3.2.1). Il répond au souci d'application distribuée dans un environnement hétérogène. Pour tout développement d'application avec ce middleware, on a à disposition le langage CA-IDL (Context Aware-Interface Description Language) qui permet d'écrire l'interface permettant de générer le code d'adaptation du logiciel. Cet intergiciel gère en effet, l'aspect d'analyse et d'adaptation au contexte.

II.3.4.9. SAFRAN

SAFRAN (Self-Adaptive FRActal compoNents) est un intergiciel créé par Pierre-Charles David lors de sa thèse [SAFRAN 2005]. Il est basé sur le modèle à composant Fractal (cf. § II.3.1.5) et permet d'associer l'adaptation lié aux composants avec la configuration dynamique programmé avec le langage FScript qui se base sur le Framework WildCat (cf. fig. 38). Il est possible, en effet, de reconfigurer l'application durant son exécution en modifiant son architecture et par voie de conséquence son comportement afin de l'adapter à son contexte. Cette adaptation est interprétée par le système comme *un*

aspect. Cette notion permet de développer l'adaptation séparément du code métier. De ce fait, il peut être intégré (tissé) dynamiquement avec les composants métier mais aussi défait (détissé) ou refait (retissé) autant de fois que l'on souhaite.

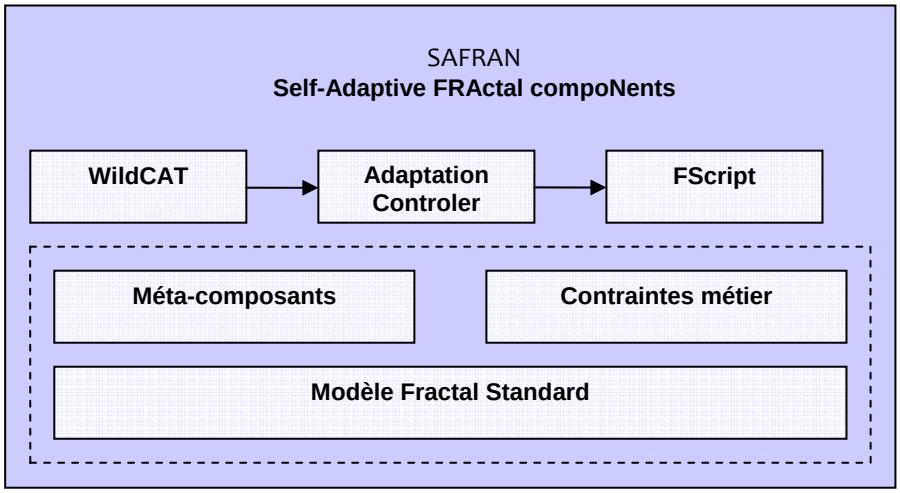

Fig. 38 : Architecture du système SAFRAN selon P.-C. David

Au niveau de l'architecture, on distingue la partie basée sur le modèle Fractal et qui permet le tissage dynamique. Elle est composée de l'entité : "Méta-composants" qui gère l'aspect adaptation lié au composant Fractal et de l'entité "Contraintes métier" qui définit des contraintes liées à une application.

La deuxième partie est constituée de l'entité : "WildCat" qui correspond au framework permettant de modéliser le contexte et l'entité : "Adaptation Controler" qui permet de traiter l'adaptation de l'application. L'entité : "FScript" correspond au langage qui permet de spécifier les configurations.

Cet intergiciel permet le développement d'applications sensibles au contexte (grâce à l'utilisation du composant Fractal) avec une forte qualité d'adaptation liée à l'utilisation de l'approche par aspect.

II.3.4.10. SOCAM

SOCAM (Service Oriented Context-Aware Middleware) est un intergiciel proposé par l'université de Singapour basé sur un modèle à service sensible au contexte [SOCAM]. L'approche est fondée sur la modélisation du contexte basée sur des ontologies. Pour ceci, le middleware est structuré en 3 couches (cf. fig. 39) suivant les rôles décrits ci-dessous :

- La couche "context sensing" est composé des capteurs qui sont chargés de remonter les informations du contexte.
- La couche "context middleware" est composé des composants "context provider" qui sont chargés de collecter le contexte et d'en faire part au composant "ContextInterpreter" qui est en charge de l'interprétation. Le composant "context

provider" de type externe recueille les informations externes au système comme par exemple la présence d'une personne, alors que celui de type interne recueille les informations concernant l'état du système telle la bande passante. Le composant "service locating" correspond aux pages jaunes.

- La couche "context application" est composée de services sensibles au contexte. Ces services s'enregistrent afin d'être informés sur la pertinence du contexte.

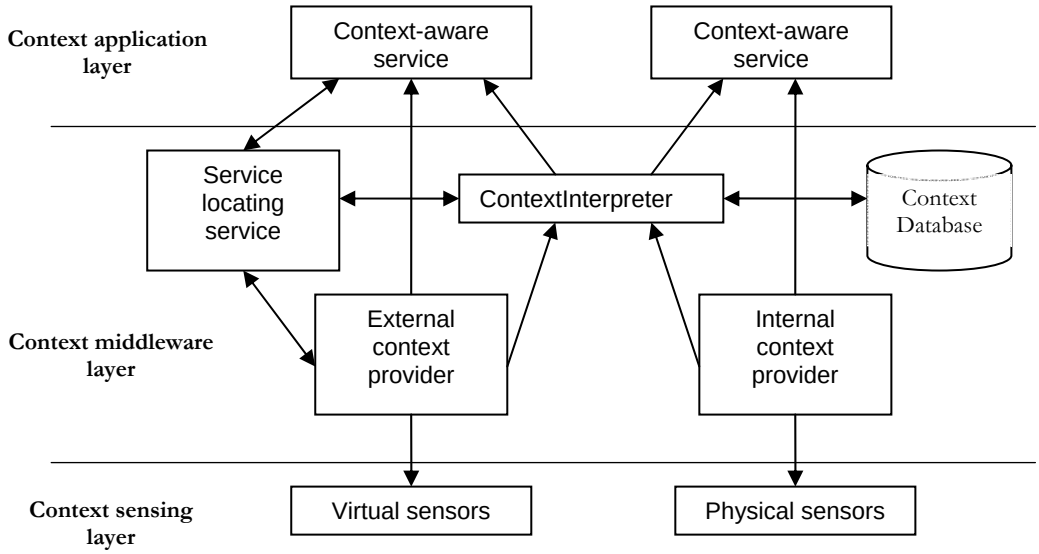

Fig. 39 : Architecture de l'intergiciel SOCAM selon T. Gu, H.K. Pung, D. Q. Zhang [SOCAM]

II.3.4.11. WComp

WComp est un intergiciel élaboré à l'Université de Nice Sophia Antipolis par l'équipe Rainbow [WComp 2009]. Il est basé sur le modèle SLCA (Service Lightweight Component Architecture) qui s'inspire du modèle à composant orienté service (cf. SCA § II.3.3.1) [WComp 2009b] mais le dépasse d'une certaine façon par l'adoption de composants légers et l'utilisation de la programmation événementielle (cf. fig. 40).

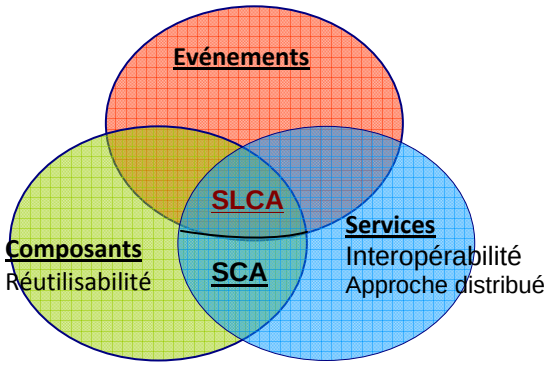

Fig. 40 : Modèle SCLC propre à WComp.

L'architecture s'appuie sur le container et le designer dont leurs rôles respectifs sont :

- pour le container, un rôle de gestionnaire et par conséquent il assure l'exécution d'un assemblage de composants légers,
- le rôle du designer est la manipulation d'application dans les containers.

Le principe est d'assembler des services dit "composites" à la manière de composant dans des containers. La plateforme propose également des dispositifs diversifiés liés à l'informatique ambiante, ce qui au final, en fait une solution complète pour la conception, le déploiement et l'adaptation des applications multidispositifs.

II.3.4.12. Conclusion

La communication d'acteurs mobiles dans un environnement réel augmenté implique de prendre en compte pour le choix du middleware, la gestion du contexte, la vérification de l'adaptabilité et l'aspect lié à la réalité augmentée. A cette fin, nous établissons les critères de comparaisons suivants :

- découverte : l'intergiciel sait détecter des dispositifs de l'environnement.
- mobilité : il doit gérer des entités mobiles.
- réactivité : le middleware peut réagir aux changements de contexte par l'utilisation d'événement ou de mécanisme de publication/souscription propre à la technologie orienté service (cf. § II.3.2 et § II.3.3).
- hétérogénéité : capacité à gérer plusieurs types de matériels, de protocoles, de systèmes d'exploitation.

Intergiciel	Découverte	mobilité	réactivité	hétérogénéité
AMIGO	✓	✓		✓
CAMidO		✓		
CARISMA		✓	✓	
CARMEN		✓	✓	
CORTEX	✓	✓	✓	✓
GAIA	✓	✓	✓	
Oxygen	✓	✓		
RSCM		✓	✓	
SAFRAN			✓	
SOCAM	✓	✓	✓	
WComp	✓	✓	✓	✓

Tableau 1 : Compararaison entre plusieurs intergiciels

Au niveau du tableau 1, Il apparait que les intergiciels CORTEX et WComp sont les seuls qui répondent à tous les critères.

Afin de les départager, nous ajoutons un autre critère qui concerne les modèles utilisés pour leur réalisation. En effet, nous avons étudié aux chapitres précédents l'avantage qu'apporte les différents paradigmes au regard de la réalité augmentée.

Intergiciel	Objet	Composant	service	Evénement	Aspect
CORTEX	✓			✓	
WComp		✓	✓	✓	✓

Tableau 2 : Comparaison entre deux intergiciels

Nous voyons que WComp a l'avantage d'utiliser quatre technologies au lieu de deux pour CORTEX.

Même si l'utilisation de la programmation objet permet la réutilisabilité qui sied à notre problème de mobilité, le modèle à composant à l'avantage de réduire la complexité de déploiement en facilitant l'interopérabilité entre les différents langages.

Nous adoptons donc WComp pour la réalisation de notre application.

CHAPITRE III

III. CONCEPTION & REALISATION

III.1. ASPECT RESEAU DE L'INTERGICIEL

III.1.1. Module XBee

Concernant la technologie ZigBee, notre choix s'est porté sur le produit OEM RF XBee Serie 2 du constructeur DIGI. Nous voulions éviter l'intrusion au niveau matériel par la programmation de microcontrôleurs par exemple, ce qui s'apparente davantage au métier d'électronicien. Ce produit évite cet écueil et est prêt à l'emploi car il est directement intégré.

Les caractéristiques techniques d'un module XBee Série 2 sont détaillées dans le tableau 3 :

Caractéristiques techniques XBee Série 2	Valeur
portée intérieur/urbain	jusqu'au 40m
portée extérieur	jusqu'à 120m
puissance d'émission R.F.	2 mW (+3dBm)
courant de repos	< 1 µA
fréquence de fonctionnement	2.4 GHz
vitesse de transmission R.F.	250 Kbps
Tension d'alimentation	2.8v à 3.4 v

Tableau 3 : Caractéristiques module XBee Série2

Pour avoir un ordre de grandeur de la taille d'un module, la figure 41 représente une image avec une pièce d'un euro et une photo d'identité à proximité de modules XBee.
Il existe 3 types de modules XBee :

- module avec antenne chip intégrée
- module avec antenne filaire intégrée

- module avec connecteur UFL

Fig. 41 : Taille d'un module XBee.

Pour notre travail, nous avons disposé d'un kit qui comprend deux cartes électroniques pouvant accueillir un module et qui peuvent être connectés par un port USB pour l'une et un port série DB9 pour l'autre.

III.1.1.1. Principes du module XBee

Un module XBee émet et reçoit des données sous forme de radio fréquence. On peut les qualifier de transceiver radio à la norme Zigbee et ont été conçu plutôt pour la réalisation de systèmes de communication dans des réseaux de capteurs. Au niveau de la récupération des données, les liaisons sont de type série. C'est pour cette raison que ces modules s'interfacent avec n'importe qu'elle circuit qui présente une liaison UART comme les microcontrôleurs par exemple. La figure 42 (cf. ci-dessous) illustre le cas d'interfaces modules s'interfacent en effet, avec des microcontrôleurs au moyen d'un port série.

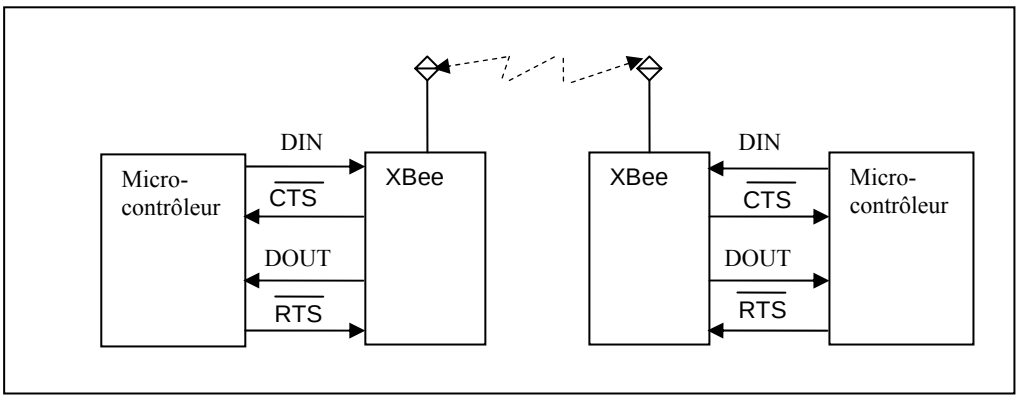

Fig. 42 : Interfaçage du module XBee

En ce qui concerne le fonctionnement interne du module XBee, il existe une broche "DIN" qui est relié à un buffer pour gérer l'aspect émission des bits et une broche

"DOUT" pour gérer la réception (cf. fig. 43). Le contrôle du flux se fait par le classique couple : CTS (Clear to Send) et RTS (Request to Send).

CTS indique que le tampon DIN est plein et que le module XBee ne peut pas pour l'instant, traiter les informations en vue de les envoyer par ondes (RF). RTS indique au tampon DOUT d'arrêter ou de laisser passer les données provenant de la borne 'Port antenne". On peut noter que lorsque le buffer DOUT est plein, les données en provenant de la borne 'Port antenne" (ondes) sont perdus.

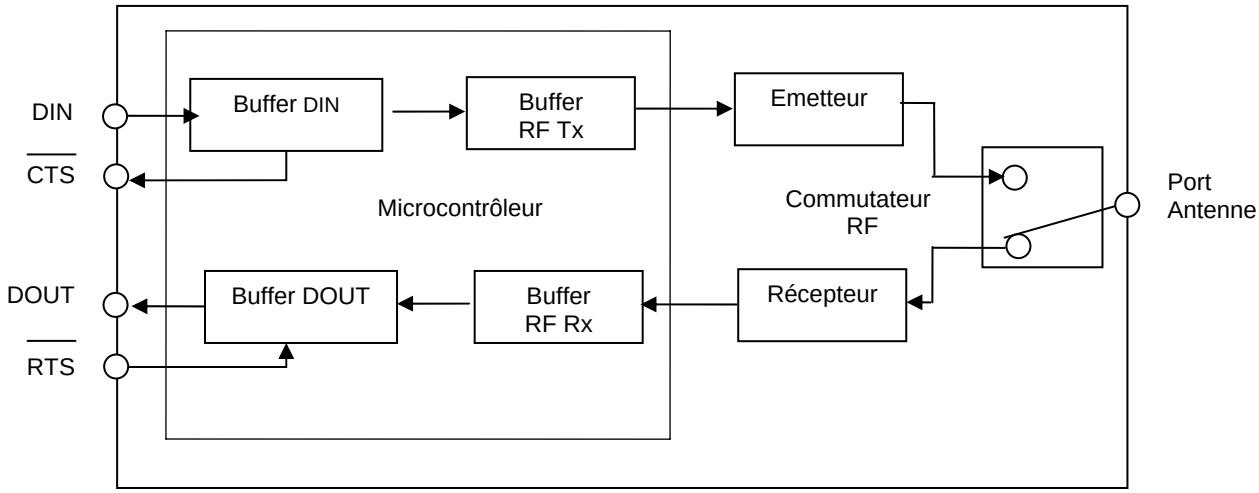

Fig. 43 : Structure interne d'un module XBee

Il existe cinq états de fonctionnement appelés communément "modes" (cf. fig. 44) :

- mode attente IDLE : correspond à l'état d'inactivité.
- mode réception (ou Rx) qui correspond à la venue de donnée "RF"[4] au niveau de l'antenne.
- mode transmission (ou Tx). Lorsque des données sont reçues au niveau du buffer DIN.
- mode sommeil permet d'économiser le courant (avantage Zigbee).
- mode commande à la réception d'un commande "AT" (cf. § suivant).

[4] RF : Radiofréquence ou fréquence radio-électrique. Correspond à la fréquence d'une onde électromagnétique rayonnée ou aux oscillations électriques correspondantes qui ont une fréquence comprise entre 9 kHz et 3000 GHz. Il peut aussi qualifier des organes électriques destiné à produire ou à réceptionner des ondes.

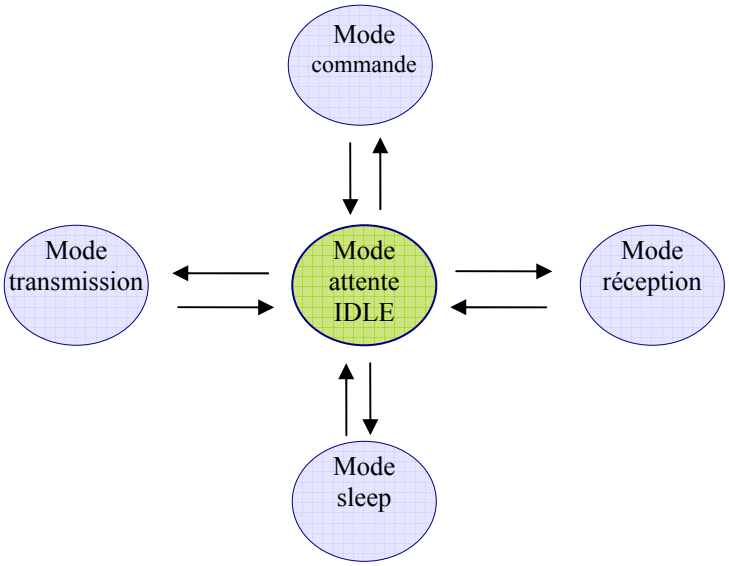

Fig. 44 : Etats de fonctionnement du module Xbee

III.1.1.2. Mode commande

Pour modifier ou lire des paramètres du module XBee, on peut le faire soit par le mode API, soit par le mode commande. Dans ce sous-paragraphe, nous nous arrêtons sur ce dernier mode qui correspond également au mode dit "AT" identique à celui des premiers modems téléphoniques. La syntaxe des commandes AT correspond au format ci-dessous :

« AT"	**+**	**"ASCII commande"**	**+**	**"Espace"** (Option)	**+**	**Paramètre** (Option, HEX) **+** **CR**

Aussi on débute toujours une communication par l'envoi de la chaine composée des trois caractères : "+++". Le module répond à travers la broche "DOUT" et il y a un "OK" qui s'affiche. Ce qui signifie que le module est prêt à recevoir des commandes au niveau de la broche "DIN". On peut ensuite utiliser les différentes commandes "at" pour connaitre les informations appropriées. On termine la communication par la commande "ATCN".

Exemple :

+++

ATDL : Lit le registre DL correspondant à l'adresse destinataire basse sur 32 bit.

ATDL 1F : Fixe la valeur de DL à 0X1F.

ATWR : Sauve les paramètres dans la mémoire volatile.

ATCN : Quitter le mode commande.

III.1.1.3. Utilitaire logiciel - X-CTU

On peut aussi configurer, lire et tester les modules XBee grâce à l'utilitaire fournit par le constructeur : "X-CTU" ou à télécharger sur son site web [DIGI 2010]. C'est un utilitaire

qui se présente avec une interface graphique fonctionnant avec des ordinateurs munis d'un système d'exploitation compatible avec au moins la version Windows 98. Ce logiciel permet entre autres choses de changer les firmwares des modules et par conséquent de les configurer. Au démarrage, on a un menu avec quatre onglets comme représenté à la figure 45.

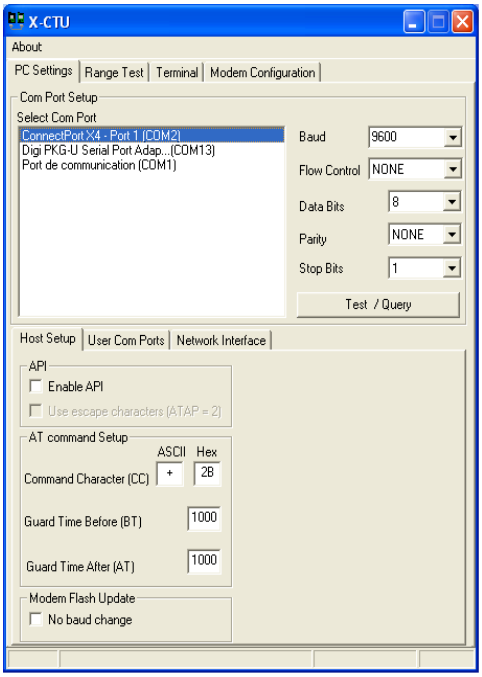

Fig. 45 : Menu principal du logiciel X-CTU.

On peut ainsi réaliser quatre fonctionnalités qu'on détaille ci-après :

- Onglet : "**PC Setting**" (cf. fig. 45). On a la possibilité de choisir tous les paramètres de communication du port série virtuel COM de l'ordinateur à lequel est connecté notre module. Par défaut, les modules XBee sont configurés pour fonctionner avec une vitesse de 9600 bauds/s, 8 bits de données, 1 bit de stop et aucune parité. On peut ensuite vérifier l'état du port grâce au bouton "Test/Query".
    - Sous-onglet : **Host Setup** : En cochant la case : "Enable API", on a la possibilité de choisir le mode API ou "commande AT" pour l'accès au firmeware.

- Onglet : **"Range Test"** (cf. fig. 46) **:** En sélectionnant cet onglet, on peut faire des essais de transmissions entre modules et vérifier ainsi la qualité du signal reçu.

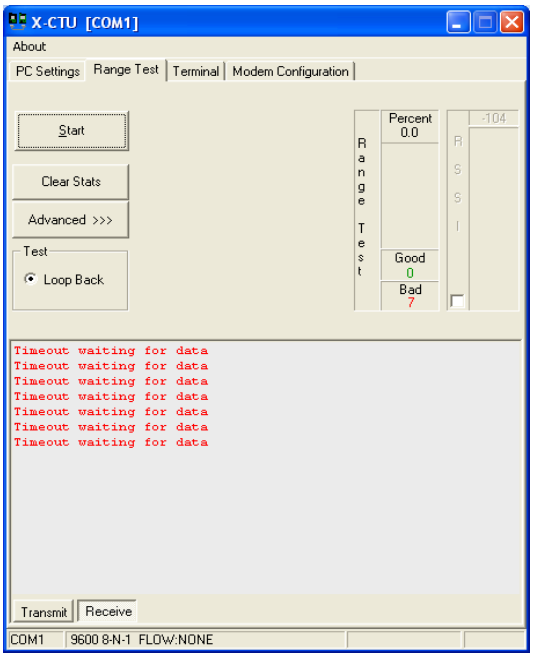

Fig. 46 : Onglet "Range Test" du logiciel X-CTU.

- Onglet : **Terminal** (cf. fig. 47) **:** Correspond à l'émulation d'un terminal qu'on peut utiliser pour dialoguer avec le module XBee, par l'utilisation de commandes "AT".

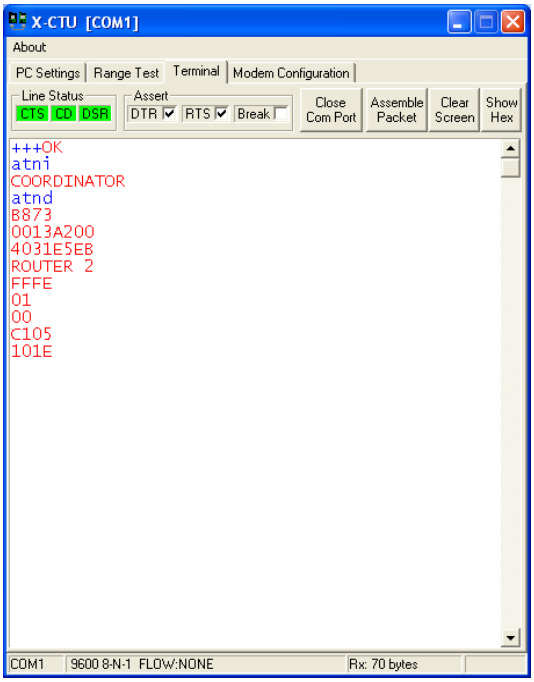

Fig. 47 : Onglet "Terminal" du logiciel X-CTU.

- Onglet : **Modem Configuration** (cf. fig. 48) **:** Cet onglet a une fonction importante car il permet de changer (ou "reflasher") le firmware et de paramétrer tous les paramètres du module XBee suivant la fonction du module et le type de réseau souhaité. Pour configurer un module, on lit, dans un premier temps, la version du firmeware et sa configuration par l'intermédiaire du bouton "Read". Ensuite on change le type de Modem, la fonction et la version pour ensuite ajuster la configuration suivant le type du réseau souhaité (cf. § suivant : configuration

réseau) et sauvegarder toute cette configuration au niveau du module par l'intermédiaire du bouton "Write".

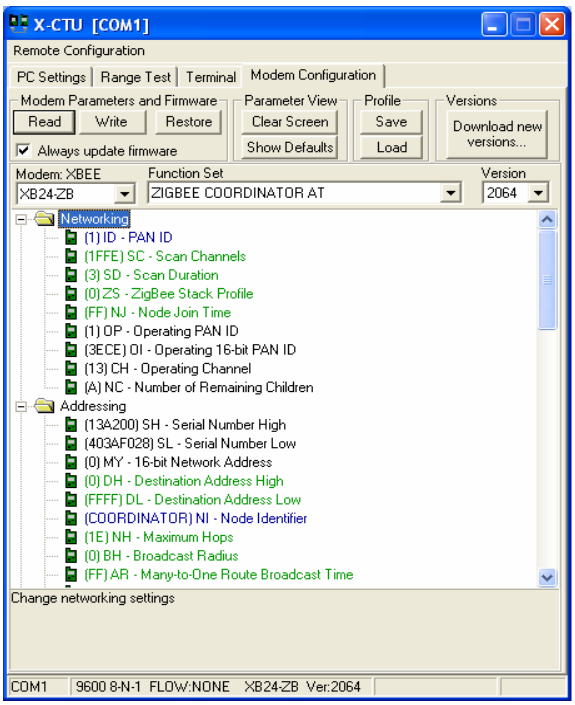

Fig. 48 : Onglet "Modem configuration" du logiciel X-CTU.

III.1.1.4. Configuration réseau

Nous avons opté pour la configuration des modules afin qu'ils puissent fonctionner en réseau maillé (mesh). L'avantage de cette typologie de réseau est que les modules sont connectés de proche en proche et de ce fait, assurent la redondance et le routage automatique. Cette typologie permet un déploiement rapide et simplifié avec une grande évolutivité de la couverture qui est un atout pour notre problématique.

D'après les recommandations du constructeur, il ne peut y avoir qu'un seul coordinateur et autant de routers ou de "end devices" que l'on souhaite.

Pour réaliser un réseau maillé, on adopte le mode broadcast. En effet, avec ce mode il n'y a pas d'accusé de réception (acknowledge) contrairement au mode unicast. Tous les modules reçoivent et acceptent les paquets de données. Pour envoyer les données sans tenir compte de l'adresse destinataire, il faut positionner l'adresse destinataire : DH = 00 00 00 00 et DL= 00 00 FF FF.

III.1.1.5. Configuration du coordinateur

Il existe deux types d'adressage possibles : un court sur 16 bits ou un long sur 64 bits.

Nous choisissons l'adressage court en configurant les paramètres importants d'un module XBee comme décrit ci-dessous :

- *ID-PAN ID* : Identificateur PAN du réseau (mesh). Il faut choisir un nombre (hexadécimal) qui soit identique pour tous les modules appartenant au même réseau. Dans notre cas, on choisit 1.
- *SH - Serial Number High* : Numéro de série du module codé sur 32 bits de poids fort. Le numéro des modules XBee (attribué au constructeur Digi) est le 13A200.
- *SL – Serial Number Low* : Numéro de série codé sur 32 bits de poids faible. C'est un numéro unique propre à chaque module. Ces numéros (SH et SL) sont imprimés par le fabriquant au dessous du module car ils forment un numéro unique équivalent à l'adresse MAC pour les cartes éthernet.
- *DH – Destination Address High* : Numéro de série du module XBee (bits de poids fort) avec lequel on désire dialoguer ou dans le cas d'un réseau maillé on met un "0".
- *DL – Destination Address Low* : Numéro de série du module XBee (bits de poids faibles) avec lequel on désire dialoguer. Dans le cas d'un réseau maillé, on met la valeur "FFFF".
- *SE – ZigBee Source Endpoint* : Configuré avec la valeur E8
- *DE – ZigBee Destination Endpoint* : Configuré avec la valeur E8
- *CI – ZigBee Cluster ID* : Configuré avec la valeur 11
- *RO – Packetization Timeout* : Nombre de caractères préparé dans le module avant de lancer la transmission. Pour qu'il n'y ait pas de latence et que tout paquet préparé soit envoyé immédiatement, il faut configurer ce paramètre avec un "0".

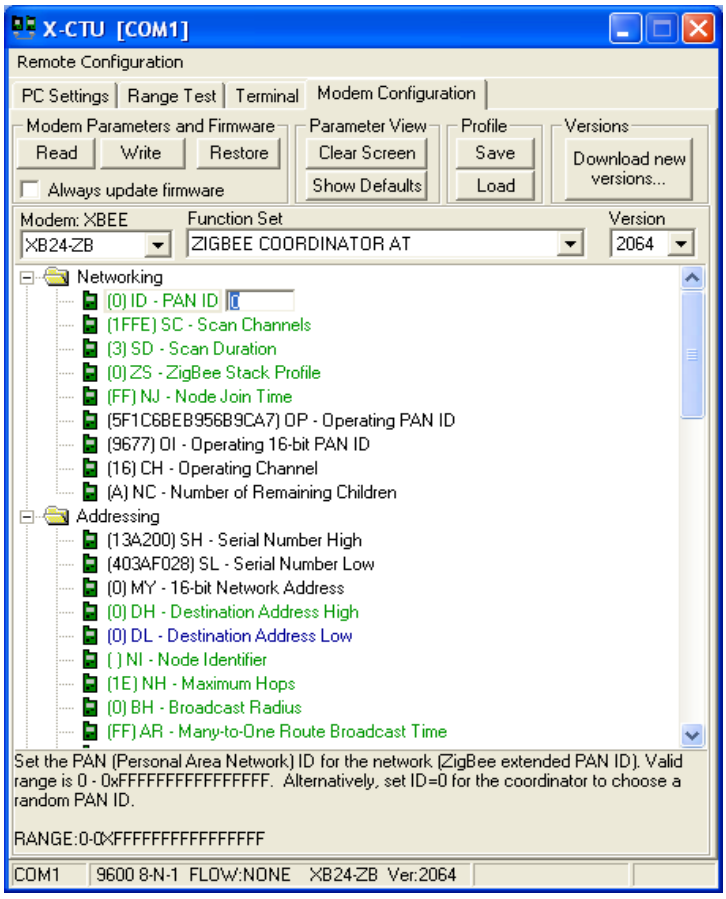

Fig. 49 : Configuration paramètre pour un coordinateur.

III.1.1.6. Configuration du router

Pour obtenir la configuration d'un router, nous avons modifié les paramètres suivants :

- *ID-PAN ID* : Identificateur PAN du réseau (mesh). Le nombre choisi (1) doit être identique à tous les modules du réseau.
- *SH – Serial Number High* : concerne les 32 bits de poids fort du numéro de série du module.
- *SL – Serial Number Low* : concerne les 32 bits de poids faible du numéro de série du module.
- *DH – Destination Address High* : Numéro de série du module XBee (bits de poids fort) avec lequel on désire dialoguer ou dans le cas d'un réseau maillé on met un "0".
- *DL – Destination Address Low* : Numéro de série du module XBee (bits de poids faibles) avec lequel on désire dialoguer. Dans notre cas, la valeur configurée est "FFFF".
- *JV – Channel Verification* : Configurez à "0" afin d'éviter un allongement de la communication.
- *SP – Cyclic Sleep Period* : La valeur est mise à 0 (doit être égale à 20 mini).
- *SE – ZigBee Source Endpoint* : Configuré avec la valeur E8.
- *DE – ZigBee Destination Endpoint* : Configuré avec la valeur E8.
- *CI – ZigBee Cluster ID* : Configuré avec la valeur 11.

- *RO – Packetization Timeout* : Nombre de caractères préparé dans le module avant de lancer la transmission. Pour qu'il n'y ait pas de latence et que tout paquet préparé soit envoyé immédiatement, il faut configurer ce paramètre avec un "0".

III.1.1.7. Configuration des "end device"

Pour la configuration d'un "end device", nous avons modifié les paramètres suivants :

- *ID-PAN ID* : Identificateur PAN du réseau (mesh). Configuré à 1 pour faire partie du réseau déjà sélectionné.
- *SH – Serial Number High* : concerne les 32 bits de poids fort du numéro de série du module.
- *SL – Serial Number Low* : concerne les 32 bits de poids faible du numéro de série du module.
- *DH – Destination Address High* : Numéro de série du module XBee (bits de poids fort) avec lequel on désire dialoguer ou dans le cas d'un réseau maillé on met un "0".
- *DL – Destination Address Low* : Numéro de série du module XBee (bits de poids faibles) avec lequel on désire dialoguer. Dans notre cas, valeur hexadecimal : "FFFF".
- *JV – Channel Verification* : Configurez à "0" afin d'éviter un allongement de la communication.
- *SP – Cyclic Sleep Period* : La valeur sera mise à 0 (doit être égale à 20 mini).
- *SE – ZigBee Source Endpoint* : Configuré avec la valeur E8.
- *DE – ZigBee Destination Endpoint* : Configuré avec la valeur E8.
- *CI – ZigBee Cluster ID* : Configuré avec la valeur 11.
- *RO – Packetization Timeout* : Nombre de caractères préparé dans le module avant de lancer la transmission. Pour qu'il n'y ait pas de latence et que tout paquet préparé soit envoyé immédiatement, il faudra configurer ce paramètre avec un "0".
- *SM –Sleep Mode :* Pour le mode de mise en veille, il faut sélectionner le choix 5 (Cyclic sleep pin- wake)

III.1.1.8. Test

Nous avons configuré les devices de façon à avoir un "coordinateur", quatre "routers" et 2 "end devices" comme indiqué ci-dessous :

coordinator	2264	FFFE	0	13A200	403D136A	0/0
router 1	2064	0	0	13A200	403AF028	0/0
router 2	2264	4729	0	13A200	4031E63A	0/0
router 3	2264	A0B7	0	13A200	4031E5F4	0/0
router 4	2264	7452	0	13A200	4031E5F3	0/0
device 1	2264	5A9C	0	13A200	4031E5EB	0/0
device 2	2264	DEC	0	13A200	4031E5DE	0/0

Tableau 4 : Configuration modules de test

En envoyant un broadcast par la commande atnd, on s'aperçoit que les modules répondent bien.

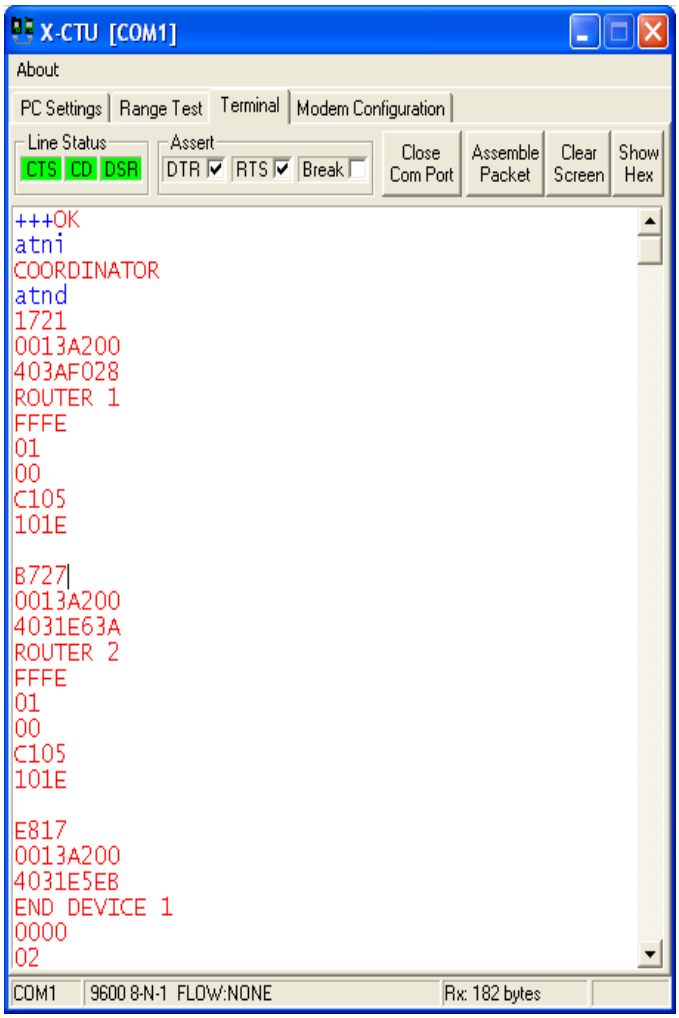

Fig. 50 : Découverte des modules par broadcast.

III.1.2. Conclusion

Nous avons configuré et testé les différents modules XBee afin qu'il puisse communiquer entre eux. De ce fait, nous avons voulu répondre à la contrainte de communication

d'acteurs mobiles en traitant l'aspect réseau qui est qu'une partie de notre travail. L'autre partie traite de l'aspect logiciel.

III.2. ASPECT LOGICIEL DE L'INTERGICIEL

Le développement se fait avec WComp afin de réaliser la couche spécifique proposant des fonctionnalités propres à la réalité mixte entre acteurs mobiles.

En effet, nous avons vu lors de notre état de l'art les avantages de WComp et qu'il était déjà conçu pour répondre à l'informatique ambiante avec laquelle la réalité augmentée partage de nombreux points communs.

III.2.1. Principe de WComp

WComp est une plateforme à composants orientée services pour informatique ambiante avec un environnement de développement rapide (RAD) qui propose des développements spécifiques. Elle permet de créer des composants pour ensuite les assembler dans différents modèles. On a la possibilité de choisir entre les modèles : "C# Bean", "C# container", "WebService Proxy" ou "UPnP Device WebServices" comme illustré à la figure ci-dessous (fig. 51).

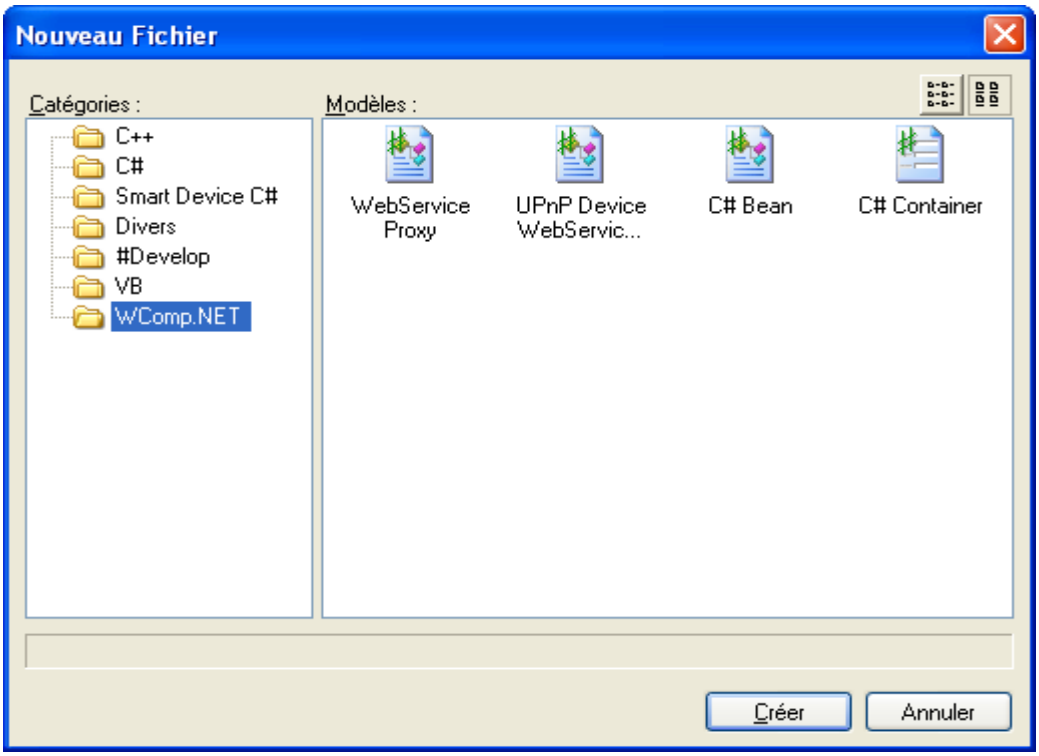

Fig. 51 : Différents types pour la création de fichier.

Cette plateforme gère des assemblages de deux types de composants. Le composant logiciel classique dont les ressources ne varient pas contrairement au composant mixte dont les ressources peuvent varier voire disparaitre. Il s'agit pour ce dernier cas de composant proxy propre au client service (Web Services et Web Services pour dispositifs).

III.2.2. SharpWComp

Bien qu'historiquement la plateforme WComp, ait été développée sur le framework Java, elle a été depuis, portée sur ".Net" (version 1.1). L'environnement choisi pour supporter cette plateforme est SharpDevelop car il correspond à un environnement de développement intégré (IDE) libre et gratuit, spécialisé pour l'environnement ".Net". SharpWcomp a été conçu comme une extension (plugin) à SharpDevelop et permet ainsi de générer du code correspondant à l'assemblage WComp, de manipuler la représentation graphique de l'application, de générer le code exécutable de l'application. Il faut aussi préciser que la version actuelle de SharpWComp correspond à la "2.0" et a été compilé avec la version 1.02a de SharpDevelop.

III.2.3. Création d'un composant

Avant de décrire la création de notre composant, nous allons d'abord exposer le principe de création d'un composant avec WComp.
Le code d'un composant se décompose en quatre parties :

- Les propriétés
- Les méthodes publiques qui seront invoquées sur le composant
- Les méthodes privées
- Les évènements levés par le composant

Lors de l'ouverture d'un nouveau fichier, on choisit le type : "C# Bean" (cf. fig. 51) pour la création d'un composant. On obtient ainsi un modèle (cf. fig. 52) à compléter qui fait appel aux notions de "delegate" et "event" pour la programmation d'événements.

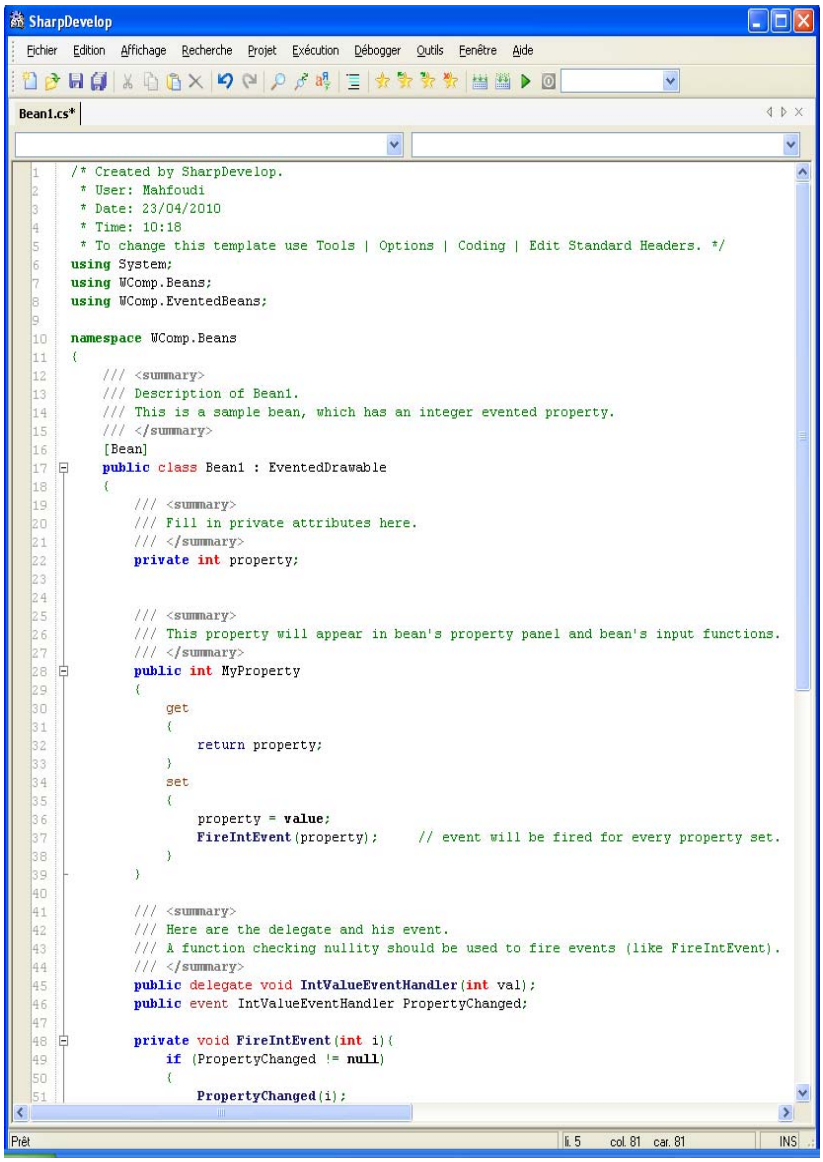

Fig. 52 : Modèle fourni par WComp pour un type C# Bean.

III.2.4. Développement composant XBee

Notre travail repose sur la gestion des modules XBee qui est basé sur la communication sans fil propre à la technologie Zigbee. De ce fait, le composant XBee a une place centrale et suivant les applications désirées, on doit développer d'autres composants à interfacer avec notre composant XBee qui propose lui de lire les données en entrée pour les restituer en sortie.

III.2.5. Composition

Cette opération se fait par l'intermédiaire du container. C'est lui en effet, qui gère à l'exécution et de façon dynamique, l'aspect d'instanciation, de désignation et de destruction des composants ainsi que l'aspect liaison entre les composants.
De façon pratique, la sélection avec Wcomp, se fait lors de l'ouverture d'un nouveau fichier (cf. fig. 51) mais cette fois, il faut sélectionner le type "C# Container#. On obtient ainsi un modèle qui par défaut est sélectionné sous le mode "Source" (cf. fig. 53). On a la possibilité de choisir deux autres modes par la sélection des onglets : "WComp .Net" ou "Design" (cf. fig. 54).

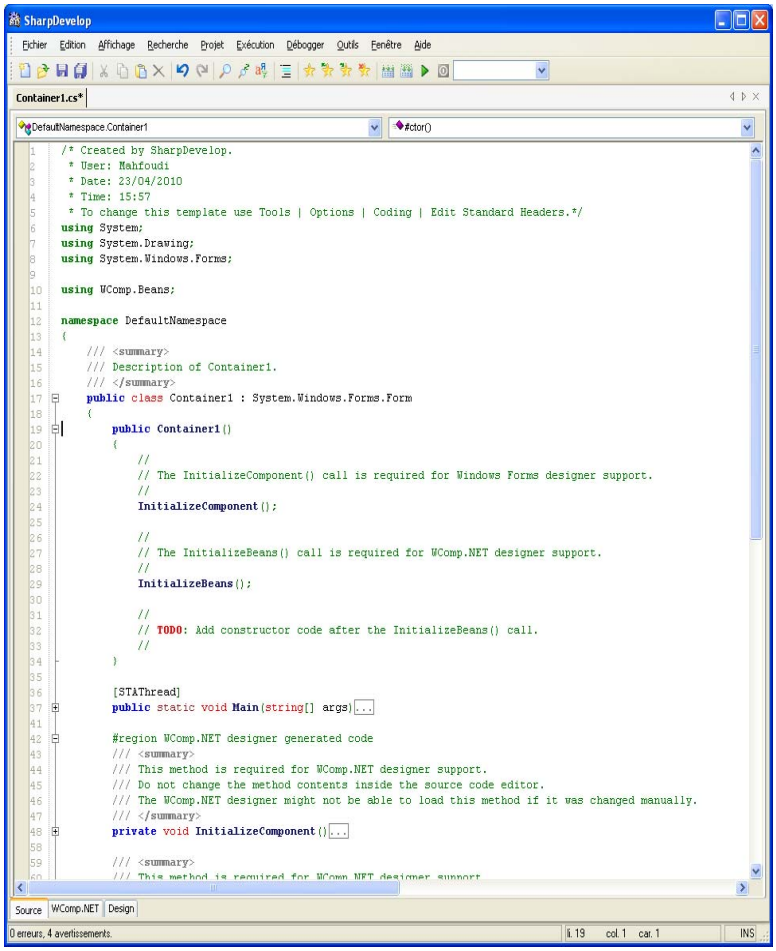

Fig. 53 : Modèle WComp pour le type "C# Container" avec le mode "source".

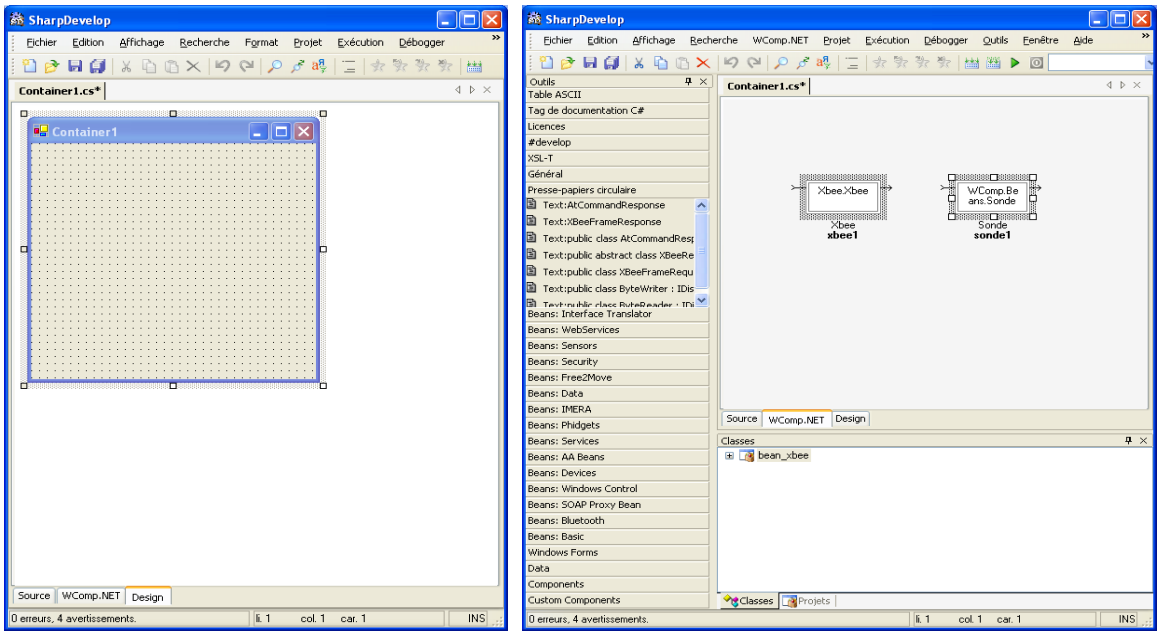

Fig. 54 : Modèle WComp pour le type "C# Container" avec le mode "Design" et " WComp .Net .

III.2.6. Travail réalisé :

Nous avons dû utiliser une librairie externe : "*MSCommlib*" car la version 1.1 du framework ".Net" ne traite pas le port série. Aussi, à la phase d'écriture du constructeur, nous avons créé l'objet "monport" grâce à l'instanciation de "MSCommClass" fournit par la librairie : "*MSCommClass*". Aussi, la configuration choisie pour notre objet "port" correspond à une vitesse de 9600 bauds, 8 bits de données, 1 bit de stop.

Pour la conception de notre composant, nous avons d'abord créé une méthode correspondant à l'émission de donnée et qui fait office d'entrée de notre composant. Les paramètres de cette méthode sont en entrée une chaine de caractère et en sortie un entier. Au départ, on ouvre le port série pour lire les données et une fois fait, on le referme par l'intermédiaire des méthodes de la classe " MSCommClass".

Pour la sortie du composant, nous utilisons un événement qui correspond aux données reçues. En effet, Wcomp nous propose un canevas (cf. Fig. 53) lors de la création d'un bean avec la signature de la fonction "*FireIntEvent" (Property)*". Cette fonction permet de lever un événement à chaque changement de la variable "*property*" qui correspond à une arrivée asynchrone des données. A cet effet, nous avons créé une méthode : "*ReceptionControl*" qui permet de surveiller l'arrivée de données au niveau du port série. Pour ceci, nous avons créé une boucle infinie grâce à la syntaxe : "while (true)" qui nous permet de garder le programme en activité. Cette syntaxe s'apparente à la programation système et pour éviter la monopolisation du microprocesseur et permettre la gestion d'autres taches, nous avons utilisés le multithreading.

Lors de l'ouverture du port série, dans la méthode "open", nous faisons appel à la gestion des exceptions par l'utilisation de "try" et "catch". Dans le cas d'une ouverture effective, du port, on fixe des variables de temps (timeout) et dans le cas contraire on retourne un false. Après l'ouverture, on peut envoyer "+++" pour passer en mode commande. Pour cela, on vérifie, que la réponse correspond bien à "OK". Sinon, on gère une exception pour lire s'il n'y a pas de données reçues.

Pour la lecture, des données, nous avons utilisé la méthode "input" de la classe "*MSCommClass*" à laquelle nous avons essayé d'attacher un tableau pour récupérer les données. Cette méthode est peu documenté et s'utilise sans paramètre. Habituellement, lors de l'utilisation de la fonction ou méthode, de la lecture d'un port série, on retrouve en premier paramètre, le fichier ou la sortie où doit s'effectuer la lecture. En deuxième paramètre, on retrouve le tampon et en troisième paramètre, le nombre d'octet à lire. Dans notre cas, la méthode "input" est relativement restreinte et offre peu de possibilités.

III.2.7. Problème rencontré

Pour la création de notre composant XBee, il a fallu gérer la communication au niveau matériel pour l'aspect du port série en plus des fonctionnalités spécifiques à ce composant. En effet, ".Net" 1.1 ne gère pas le port série contrairement aux versions supérieures ou égale à la version 2. Le traitement de cet aspect s'apparente à la création d'un driver et la tache a été ardue et n'a pas donné de résultat exploitable.

III.2.8. Actions entreprises

- **Première démarche**

Pour contourner ce problème, notre démarche a été dans un premier temps, d'utiliser le projet OpenSource : Networking Toolkit [Networking Toolkit 2009] intitulé Zigbee qui utilise le module XBee. Ce projet a été développé sous le framework ".Net" 3.5 avec le langage C#. Aussi, nous avons compilé ce code pour ".Net" 2 afin d'obtenir deux fichiers "dll" et les utilisés avec SharpDevelop pour créer notre composant.

Pour ceci, nous avons changé la configuration de SharpDevelop afin qu'il puisse incorporer des fichiers "dll" sous ".Net" 2 et créer notre composant sous ".net" 1.1 (ce qui n'est pas changeable). A l'essai, WComp n'acceptait pas aussi les fichiers dll compilé avec cette version de ".Net".

- **Deuxième démarche**

Nous avons réalisé un programme en C# sous ".Net" 2 pour vérifier notre scénario N° 1. Nous avons ainsi simulé le thermomètre par la saisie des données à partir d'un PC et

avons vérifié que les informations arrivent bien à l'autre module grâce à l'affichage de ces mêmes données sur un ordinateur portable (cf. fig.67).

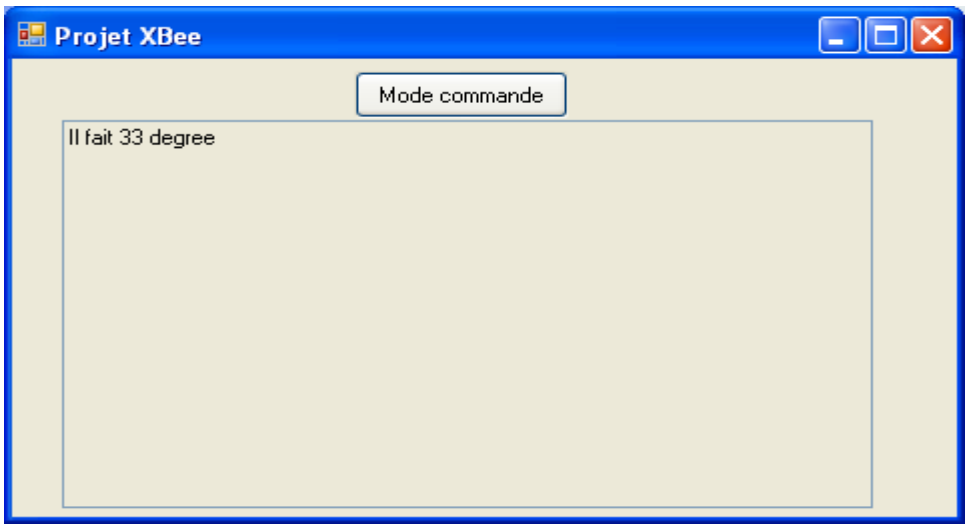

Fig. 55 : Affichage au niveau des deux machines distantes.

Fig. 56 : Test avec deux machines distantes.

- **<u>Troisième démarche</u>**

Nous avons contacté l'équipe Rainbow pour leur demander la version 2.0 de Wcomp. Messieurs Tigli et Lavirotte nous ont aimablement et exceptionnellement fourni la version béta. Nous l'avons malheureusement obtenue le dernier jour de mon stage et n'avons pas eu l'heur de la tester.

CHAPITRE IV

IV. ETUDES DE CAS

Le but est d'expérimenter l'approche de conception qui vise la mise en place de notre middleware qui implémente les concepts de la communication d'acteurs mobiles dans un environnement réel augmenté. Cette approche se structure suivant deux aspects bien distincts concernant d'une part le réseau et d'autre part le développement logiciel.

IV.1. SCENARIOS

Nous élaborons deux scénarios afin d'illustrer notre approche.

Le premier porte sur l'envoi de température et le second sur le guidage d'aveugle dans le métro.

IV.1.1. Scénario : envoi température

Le scénario choisi correspond à l'envoi de la température extérieure suivant la demande afin d'illustrer notre contrainte de réalité augmentée. Aussi, une personne peut avoir une impression sur le temps mais ne pas connaitre exactement la température. Elle souhaite interroger directement le thermomètre qui est situé à l'extérieur pour avoir une idée plus précise. L'information lui est alors envoyée grâce au réseau sans fil sur son poste et/ou son PDA. Ce premier exemple illustre la problématique de réalité mixte pour un acteur mobile.

IV.1.1.1. Cas d'utilisation

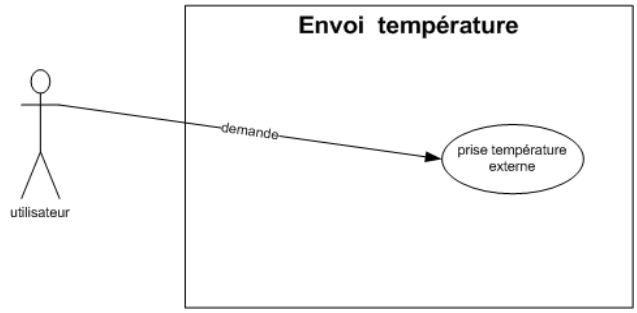

Fig. 57 : Cas d'utilisation envoi température

L'acteur peut demander la température à partir de son ordinateur s'il est immobile ou il peut être mobile et demander l'information à partir de son PDA.

IV.1.1.2. CoCSys

- **Phase 0** : Collecte des besoins

  - Avant de rentrer en réunion, l'utilisateur a constaté un temps couvert alors qu'il a pris soin de se renseigner auprès de Meteo France qui a annoncé un temps ensoleillé avec température de saison. Quelques instants avant la fin de la réunion, notre utilisateur hésite à aller récupérer sa veste au bureau car il est pressé et doit respecter son rendez-vous. Il souhaiterait interroger le thermomètre extérieur pour l'aider à prendre une décision.
  - Un père de famille se rend compte qu'il fait un vent glacial et pour avoir un ordre d'idée, il interroge le thermomètre en sachant pertinemment qu'il faudra rajouter 2 degrés correspondant à la température ressentie et conseiller son fils de bien se couvrir lorsqu'il sortira.

- **Etape 1** : écriture des scénarios

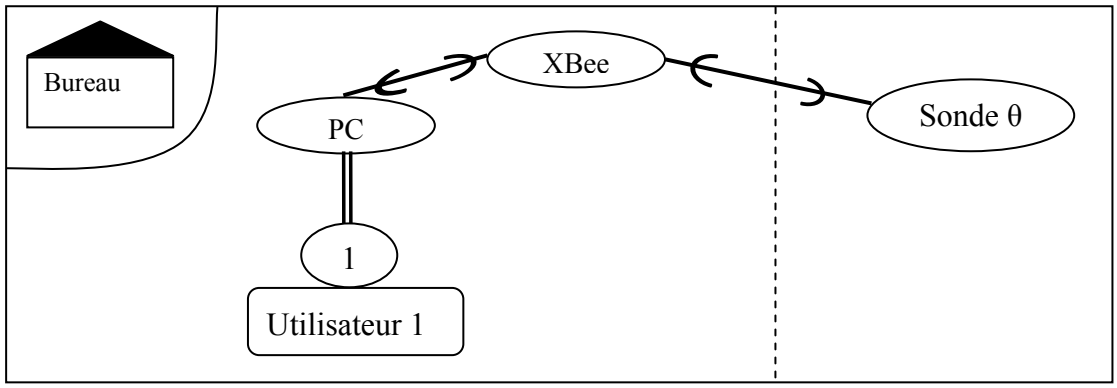

Fig. 58 : Scénario contextualisé n°1.

- **Etape 2** : compléter les scénarios

Dans notre cas, on peut considérer que le scénario est complet et dû au fait qu'il est relativement simple.

- **Etape 3** : Identifier le but

Le scénario permet de donner la température extérieure suivant la demande de l'utilisateur.

- **Etape 4** : Identifier les scénarios manquants

Nous avons relevé le cas où c'est l'utilisateur qui demande la température. On a oublié que le contexte peut aussi apporter des informations lorsque par exemple la température franchit des paliers (-5°c, 0°c, 10°c, 20°c,…).

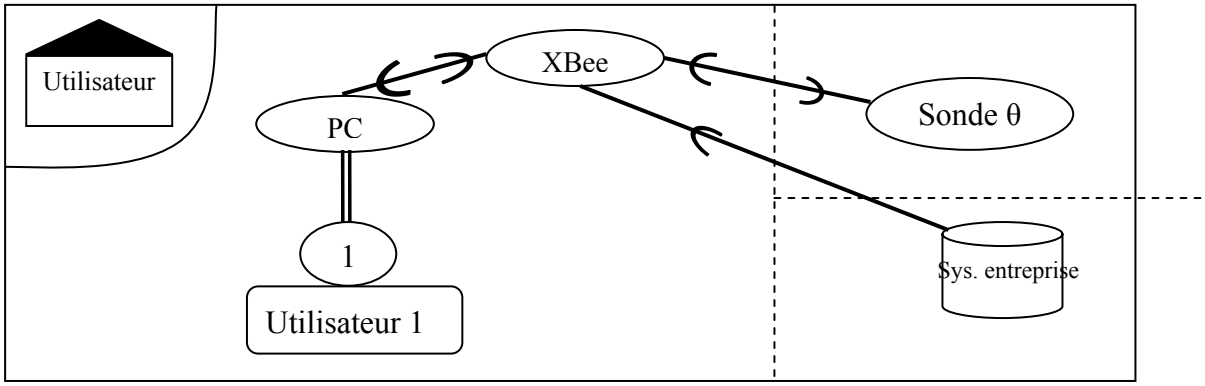

Fig. 59 : Scénario contextualisé n°2 rajouté après réflexion.

- **<u>Phase 1</u>** : <u>Construction du modèle comportemental</u>

- **Etape 1** : compléter les scénarios

A cette étape, on organise les informations des différents scénarios pour réaliser un modèle comportemental global.

Scénario contextualisé n°1 : utilisateur demande la température.

Scénario contextualisé n°2 : température donnée après franchissement de valeurs de façon automatique.

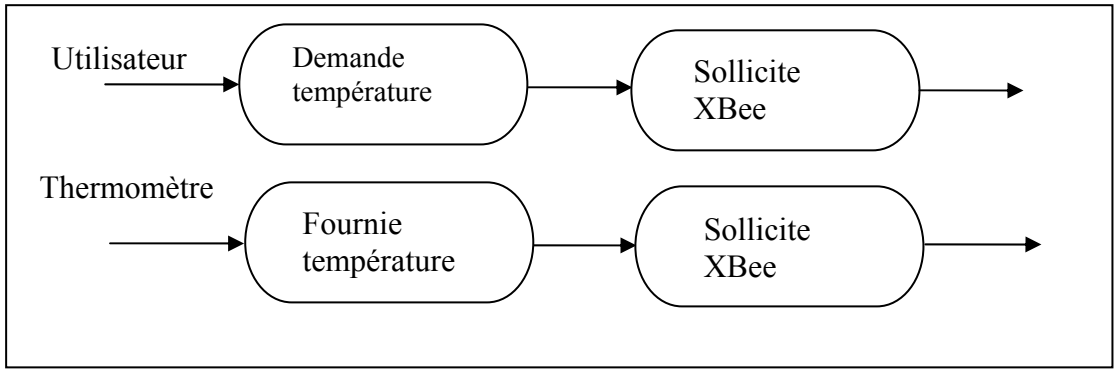

Fig. 60 : organisation des scénarios pour le modèle comportemental.

➢ **Etape 2** : extraction des informations

L'extraction se fait par la décomposition des scénarios. Le résultat est mis dans les catégories idoines comme rôle, acteur, activité, tâche, artefact,… .

Il faut ensuite trier les tâches pour savoir si elle est de type production, communication ou coordination.

o **Décomposition du scénario contextualisé n°1**

Contexte local
Spatial : Bureau, déplacement (nomade) Evénementiel : Demande température Dispositif : PC, PDA

Tâches
Demande température Envoie ordre à modules XBee Lecture thermomètre Transmission valeur

Rôles
Utilisateur

Artefacts
Réseaux : module XBee Serveur

Processus
Demande température

Taches de production (expression des tâches utilisateur)	**Taches d'organisation** (expression de la régulation et de la coordination	**Tâches de communication** (expression des échanges entre utilisateur et système)
Demande température	Sollicite XBee	Lecture température

Artefacts de production	Artefacts d'organisation	Artefacts de communication
	PC utilisateur Thermomètre	Module XBee

Décomposition du scénario n°1

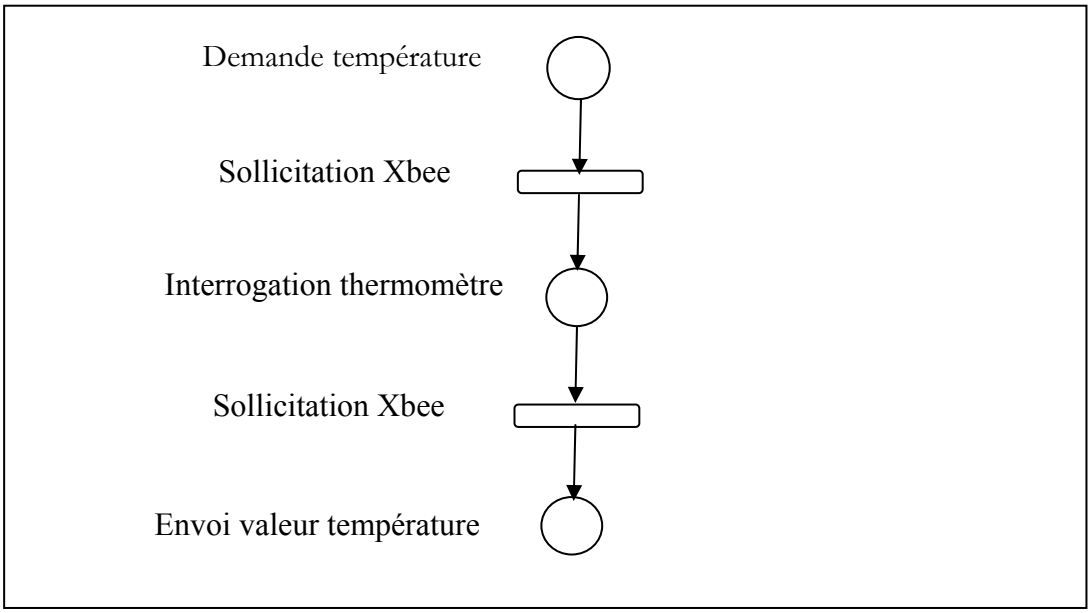

Fig. 61 : Décomposition scénario contextualisé n° 1 envoi température.

- **Diagramme de tâches du scénario n°1**

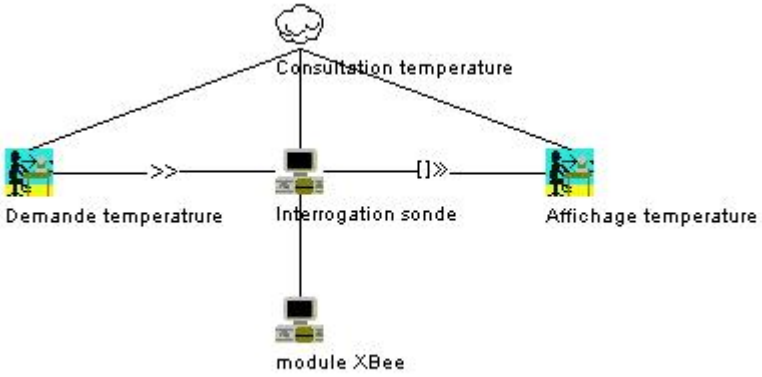

Fig. 62 : Modèle de tâches pour scénario n°1.

- **Décomposition du scénario contextualisé n°2 :**

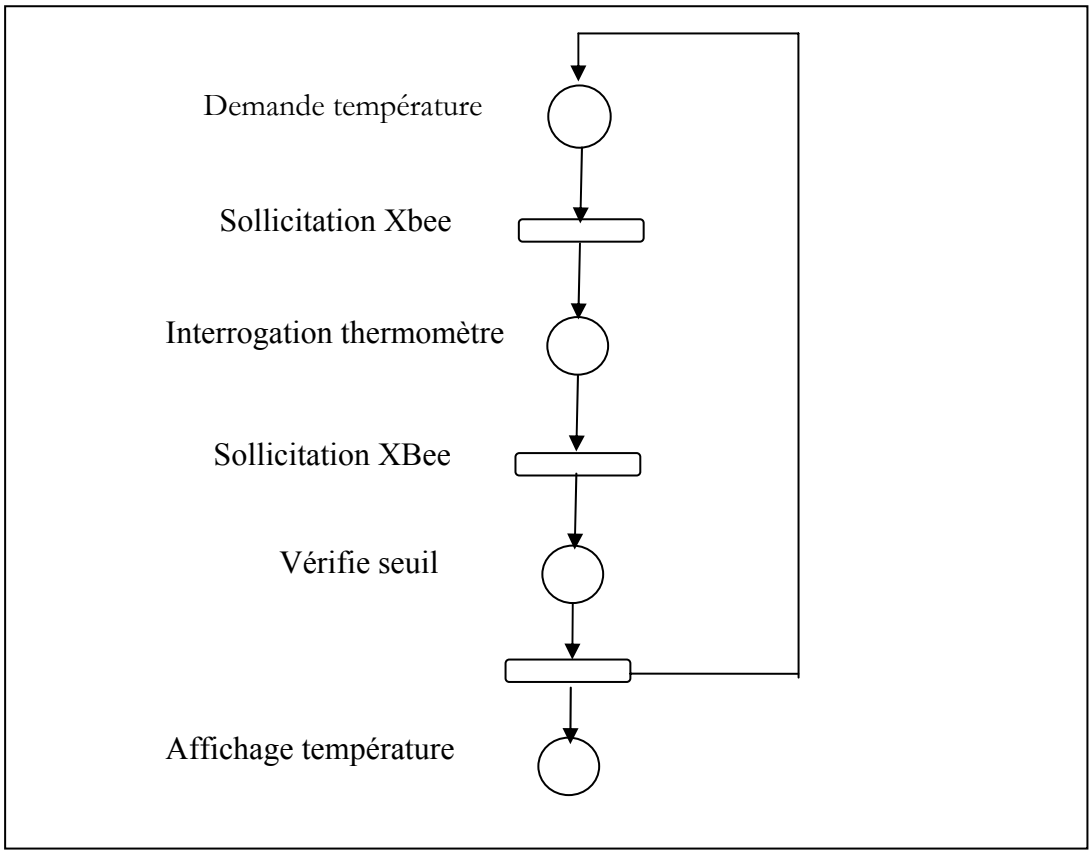

Fig. 63 : Décomposition scénario contextualisé n° 2 envoi température.

- **Diagramme de tâche du scénario n°2**

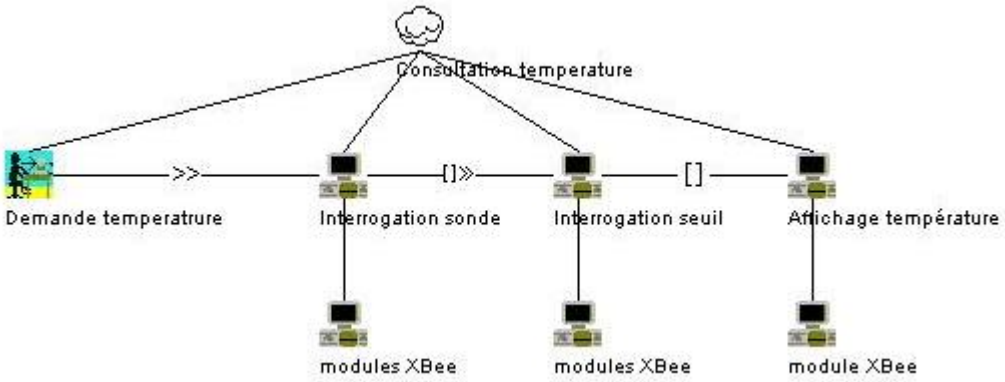

Fig. 64 : Modèle de tâches pour scénario n°2.

- **Modèle de rôles :**

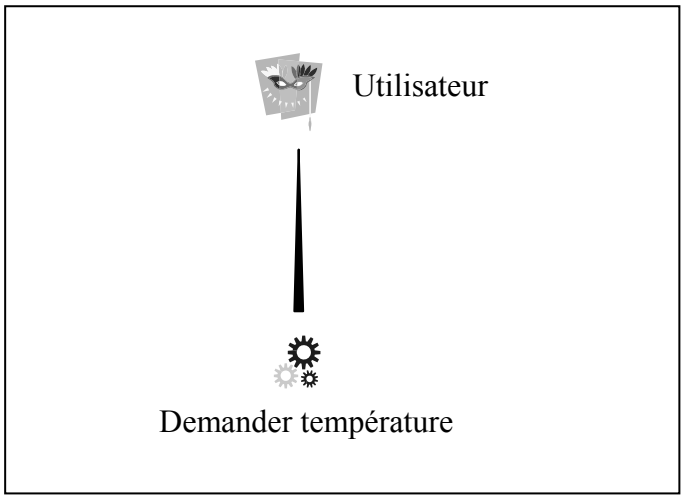

Fig. 65 : Modèle de rôle

Le modèle de rôle pour les scénarios 1 et 2 est relativement succinct mais nous n'oublions pas que nous sommes dans un environnement de réalité mixte ou le contexte peut jouer un rôle. En effet, dans notre cas, la température peut être affichée non pas, par la demande de l'acteur mais suivant le seuil qu'elle atteint.

IV.1.1.3. Modèle IRVO

La figure ci-dessous (fig. 66) illustre la modélisation en IRVO de l'aspect de réalité augmentée. Le thermomètre par l'information qu'il apporte participe à l'augmentation de la réalité et est représenté en conséquence à cheval entre le monde réel et virtuel. La partie représentée dans le monde réel correspond à la prise de température par la sonde qui correspond bien à un acte effectif de la réalité environnante. La partie du monde virtuel correspond au dispositif de liaison qui assure la transmission des informations qui participent eux, à enrichir la perception de l'utilisateur. De ce fait, il participe bien du monde virtuel.

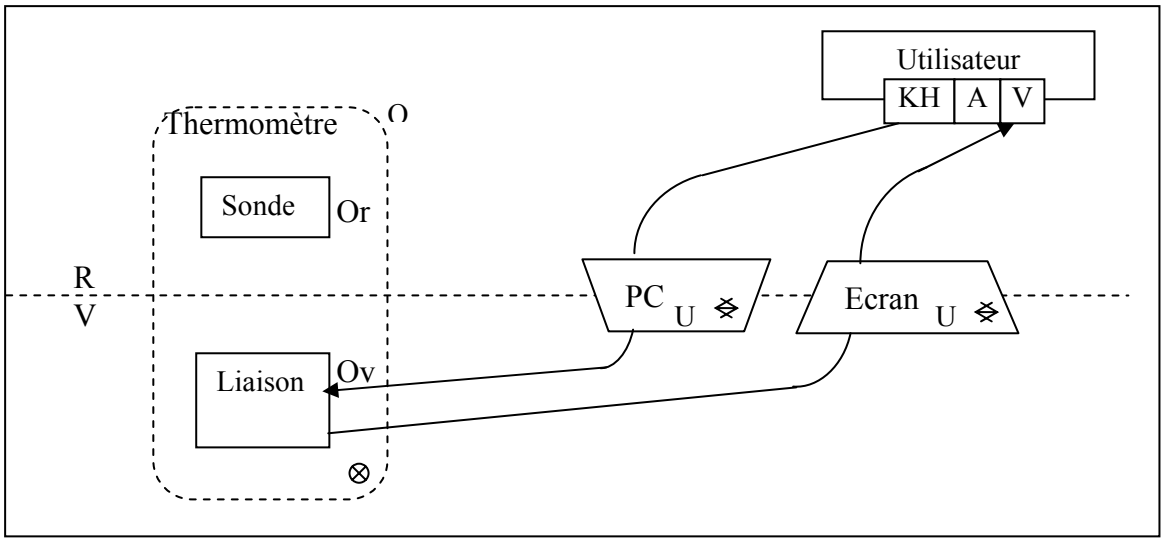

Fig. 66 : Modèle d'interaction IRVO

IV.1.2. Scénario : Guidage d'aveugles à travers le métro

Ce scénario illustre l'aspect de réalité augmentée ainsi que l'aspect de communication entre acteurs mobiles. Il correspond en effet, au guidage d'aveugle à travers le métro. La personne déficiente visuelle est munie d'un module XBee pour assurer la communication sans fil. Elle est aussi munie d'un dispositif de lecteur RFID qui fait office de capteur. Suivant la direction qu'elle prend, les modules transmettent l'information et suivant les modules sollicités on peut savoir quelle direction prend la personne. Elle peut être munie d'un dispositif audio qui lui apporte l'information pour la guider voire la corriger. On souhaite aussi l'informer de la présence d'un autre voyant lorsqu'une demande est faite simultanément.

L'idée est de former un réseau maillé avec les modules XBee, alimentés avec 2 piles de 1,5 v. et positionné à des emplacements appréciés pour couvrir l'espace de déplacement des personnes. Ces modules fixes font office de capteurs en quelques sortes et communiquent avec les modules mobiles transportés par les personnes.

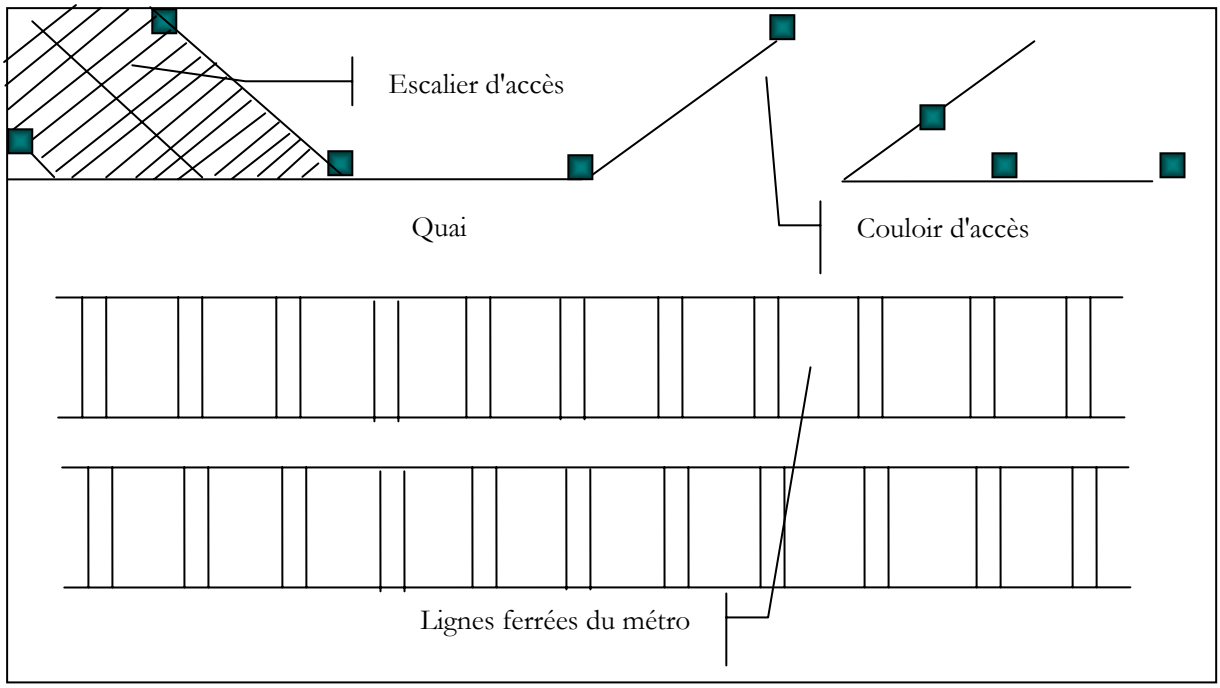

Fig. 67 : Positionnement des modules XBee

IV.1.2.1. Cas d'utilisation

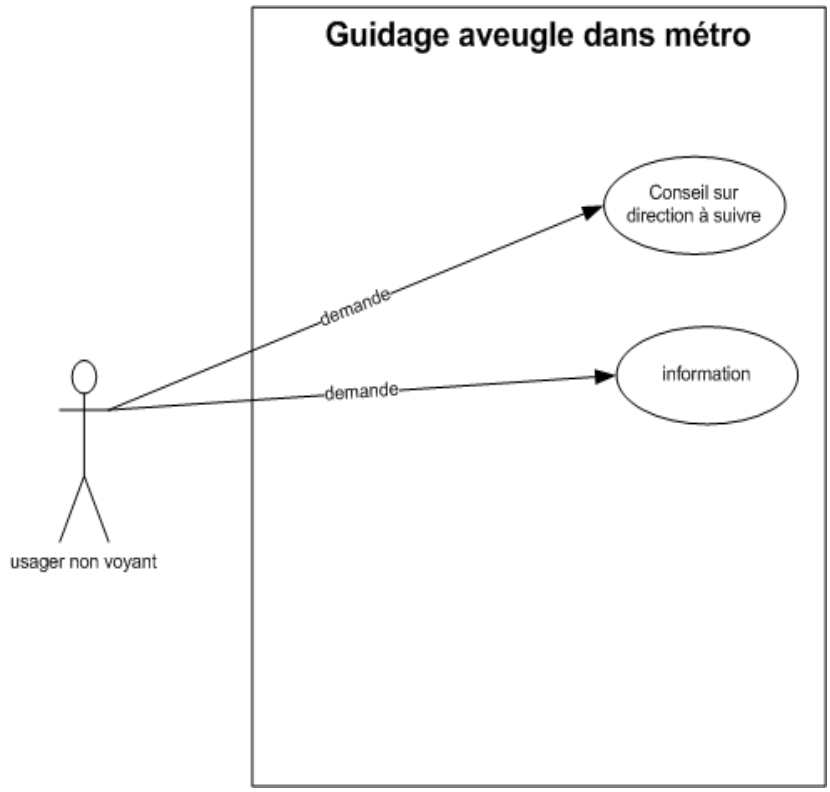

Fig. 68 : Cas d'utilisation guidage.

IV.1.2.2. CoCSys

- **Phase 0** : Collecte des besoins

  La plupart des stations de métro lyonnais ont l'avantage de ne pas être trop enterrées en profondeur et d'avoir un réseau de couloirs relativement simple comparé à celui de Paris. Mais ce n'est pas le cas de toutes les stations et particulièrement, celles avec des croisements de lignes. Ceci à une conséquence sur la densité des modules à installer. Pour un usager mal voyant, le guidage répond à un vrai besoin quelque soit la configuration des stations. Cet usager a besoin d'être informé s'il l'est ou pas dans la bonne direction. On propose qu'il soit muni d'un dispositif audio pouvant l'informer d'incident particulier.

  - **Etape 1** : écriture des scénarios

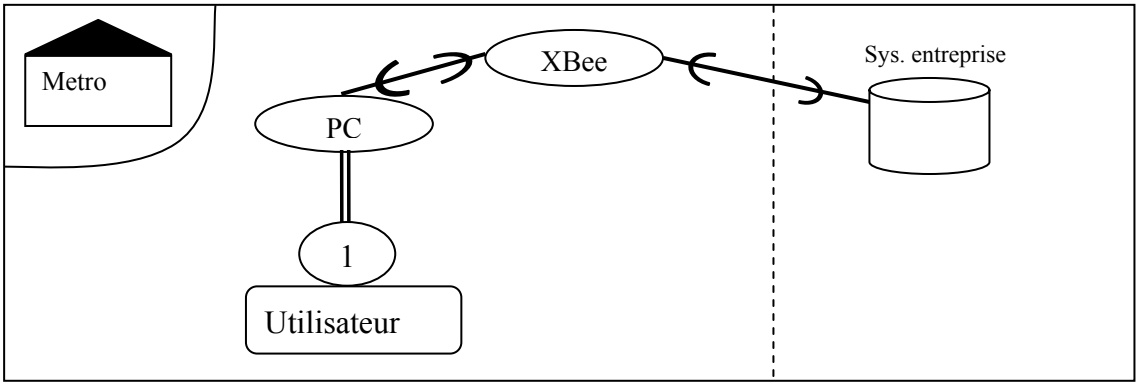

Fig. 69 : Scénario contextualisé .pour guidage aveugle

  - **Etape 2** : compléter les scénarios

Pas de rajout particulier à ce niveau là.

  - **Etape 3** : Identifier le but

Le but est d'apporter une information supplémentaire à un usager non voyant qui par définition est dépourvue d'information visuelle.

  - **Etape 4** : Identifier les scénarios manquants

Nous n'avons pas à priori de scénarios à rajouter.

- **Phase 1** : Construction du modèle comportemental

  - **Etape 1** : compléter les scénarios

  Afin de réaliser un modèle comportemental global, on peut rajouter qu'en fonction de la direction prise, il s'agit d'émettre des sons qui correspondent soit à la bonne direction, soit à une mauvaise, soit à un son qui correspond à une attente d'information.

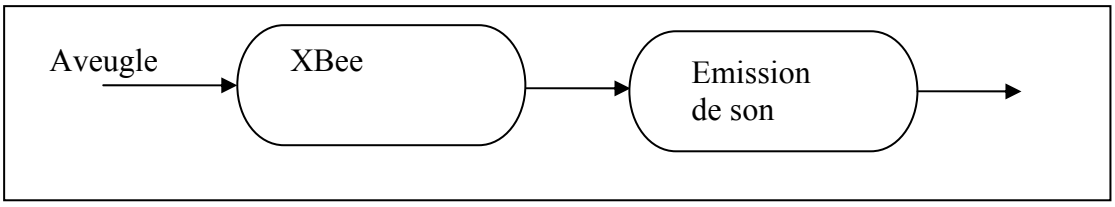

Fig. 70 : organisation des scénarios pour le modèle comportemental.

➢ **Etape 2** : extraction des informations

L'extraction se fait par la décomposition des scénarios. Le résultat est mis dans les catégories idoines comme rôle, acteur, activité, tâche, artefact,… .

Puis on trie les tâches pour savoir si elles sont de type production, communication ou coordination.

o **Décomposition du scénario contextualisé :**

Contexte local
Spatial : déplacement métro Evénementiel : indique direction Dispositif : module XBee

Tâches
Envoi id module Réception information

Rôles
Usager non voyant
Processus
envoi information

Artefacts
Réseaux : module XBee Serveur

Taches de production (expression des tâches utilisateur)	**Taches d'organisation** (expression de la régulation et de la coordination	**Tâches de communication** (expression des échanges entre utilisateur et système)
Demande information	Sollicite XBee	Envoi information

Artefacts de production	**Artefacts d'organisation**	**Artefacts de communication**
	Serveur entreprise	Module XBee

Décomposition du scénario

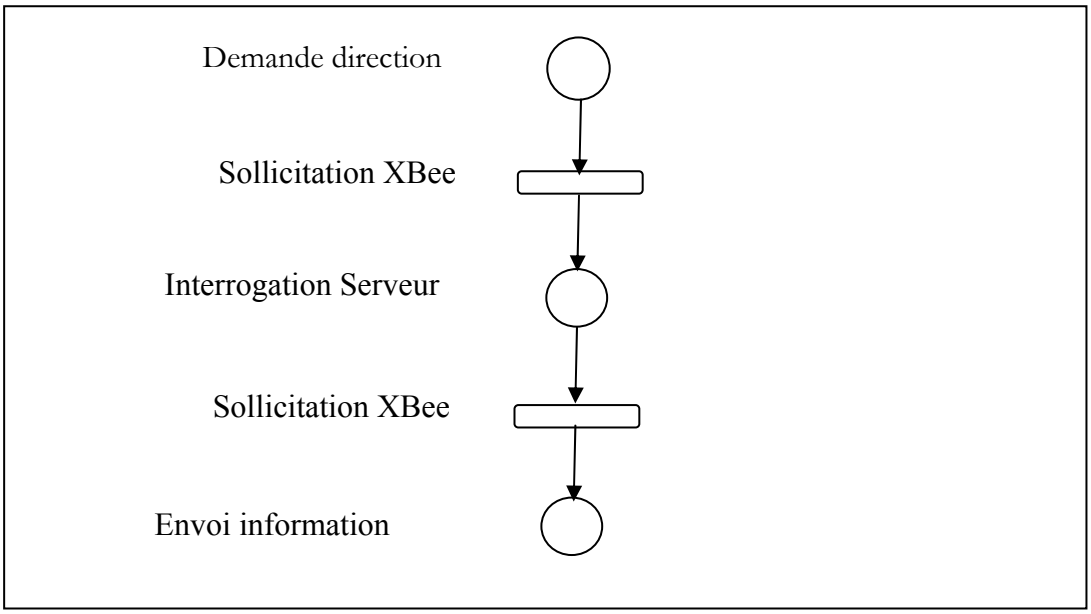

Fig. 71 : Décomposition scénario guidage aveugle.

IV.2. Conclusion

Le premier scénario a permis d'illustrer notre problématique de réalité augmenté pour un acteur mobile. Au préalable, l'information concernant la température ambiante envoyée sur le PDA d'une personne soit à sa demande soit automatiquement en fonction du franchissement de palier déterminé. Une fois, la température connue, notre utilisateur pourra la transmettre à une autre personne. Cet aspect de communication entre acteurs mobiles est particulièrement illustré avec le second scénario. En effet, le guidage de personnes non voyantes à travers les couloirs du métro met davantage l'accent sur le rôle du réseau.

CHAPITRE V

V. CONCLUSION

Notre étude a exploré des domaines distincts comme les réseaux, la Réalité Mixte (et/ou l'informatique ubiquitaire), les middlewares afin de répondre à des problématiques propres à la Réalité Augmentée et plus précisément dans un cadre collaboratif.

Dans un premier temps, nous avons étudié les concepts et les outils liés à la Réalité Augmentée et particulièrement ceux développé et utilisé dans l'équipe du laboratoire LIESP de l'Ecole Centrale de Lyon.

Dans la perspective de notre problématique, nous avons opté pour la technologie Zigbee afin d'assurer l'aspect collaboratif et également pour l'avantage qu'elle procure. En effet, cette technologie est la moins gourmande en énergie électrique des technologies sans fil.

Aussi, nous avons étudié les produits existants sur le marché et avons opté pour les modules XBee. Ces modules ont l'intérêt d'être prêts à l'utilisation et évitent ainsi l'écueil de la programmation des puces électroniques comme le propose la majorité des autres produits du marché.

Pour la réalisation de l'intergiciel, nous avons étudié les paradigmes et les plateformes susceptibles d'être utilisé et avons aussi réalisé un état de l'art de ceux déjà existants.

Suite à cette étude, notre choix s'est arrêté sur WComp qui a l'avantage d'être celui qui répond à notre besoin. Nous avons exploité l'aspect d'environnement de développement rapide (*RAD*) pour composant orienté service afin d'apporter notre contribution à la mise en place d'un composant spécifique répondant au besoin de communication réseau entre acteurs mobiles propre au module XBee.

ANNEXES

Annexe 1 : Configuration des modules XBee

Annexe 1.1 : Configuration coordinator

Modules XBee : XB24-ZB			
ZigBee coordinator at			
Networking		Serial Interfacing	
Id – Pan id (tous les modules à la même valeur)	**1**	BD Baud Rate	3 9600
SC – Scan channels	1FFE	NB Parity	0
SD– Scan Duration	3	SB Stop Bits	1
ZS - ZigBee Stack profile	0	RO Packetization Timeout	3
NJ – Node Join Time	FF	D7 – DI07 Configuration CTS Flow Control	0
OP Operating PAN ID (1)	-	D6 – DI06 Configuration	0 disable
OI Operating 16 bit PAN ID (3ECE)	-	**AT Command Options**	
CH Operating Channel (14)	-	CT AT command Mode Timeout	64
NC Number of Remaining Children (9)	-	GT Guard Times	3E8
Addressing		CC Command Sequence Character	2B
SH Serial Number High (**13A200**)		**Sleep Modes**	
SL Serial Number Low (**403AF028**)		SP Cyclic Sleep Period	20
MY 16 bit Network Address(**0**)		SN Number of Cyclic Sleep Periods	1
DH Destination Address High	0	I/O Setting	
DL Destination Address Low	FFFF	D0 – AD0/DIO0 Configuration 1= Commissioning button	1
NI Node Identifier	(coordinator)	D1 – AD1/DIO1 Configuration 0=disabled	0
NH Maximum Hops	1E	D2 – AD0/DIO2 Configuration	0
BH Broadcast Radius	0	D3 – AD0/DIO3 Configuration	0
AR Many to One Route Broadcast Time	FF	D4 – AD0/DIO4 Configuration	0
DD Device Type Identifier	30000	D5 – DIO5/Assoc Configuration 1=Associated indicator	1
NT Node Discovery Backoff	3C	P0 – DIO10/PWM0 Configuration	1
NO Node Discovery Options	0	P1 – DIO11 Configuration	0
NP Maximum Number of Transmission Bytes (54)		P2 – DIO12 Configuration	0
ZigBee Addressing		PR Pull up Resistor Enable	1FFF
SE ZigBee Source Endpoint	E8	LT Associate LED Blink Time	0
DE ZigBee Destination Endpoint	E8	RP RSSI PWM Timer	28
CI ZigBee Cluster ID	11	**I/O Sampling**	

RF Interfacing		IR IO Sampling Rate	0
PL Power Level	4 (highest)	IC IC Digital IO Change Detection	0
PM Power Mode (1= boost mode enabled)	1 boost	V+ Supply Voltage High Threshold	0
Security		**Diagnostic Commands**	
EE Encryption enable	0 disabled	VR (2064) Firmware Version	
EO Encryption Options	0	HV (1944) Hardware Version	
KY Encryption Key	( )	AI (0) Association Indication	
NK Network Encryption Key	( )	DB (3A) RSSI of Last Packet	
		%V (A90) Supply Voltage	

Annexe 1.2 : Configuration router

	VR	MY	ID	SH	SL	DH/DL
Module XBP24	2264	FFFE	234a	13A200	403D136A	0/0

Modules XBee pro : XBP24-ZB			
ZigBee Router at			
Networking		**Addressing**	
Id – Pan id (tous les modules à la même valeur)	**1**	SH Serial Number High (13A200)	
SC – Scan channels	1FFE	SL Serial Number Low (<N° unique>)	
SD– Scan Duration	3	MY 16 bit Network Address(BFA1)	
ZS - ZigBee Stack profile	0	DH Destination Address High	0
NJ – Node Join Time	FF	DL Destination Address Low	0
NW – Network Watchdog Timeout	0	NI Node Identifier	router
JV – Channel Vérification	**0** **disabled**	NH Maximum Hops	1E
JN – Join Notification	0 disabled	BH Broadcast Radius	0
OP Operating PAN ID (FFD25DFC279EC686)	-	AR Many to One Route Broadcast Time	FF
OI Operating 16 bit PAN ID (48C1)	-	DD Device Type Identifier	30000
CH Operating Channel (14)	-	NT Node Discovery Backoff	3C
NC Number of Remaining Children (C)	-	NO Node Discovery Options	0
RF Interfacing		NP Maximum Number of Transmission Bytes (54)	
PL Power Level	4 Highest	ZigBee Adressing	
PM Power Mode (1= boost mode enabled)	1 boost	SE ZigBee Source Endpoint	E8
Security		DE ZigBee Destination Endpoint	E8

EE Encryption enable	0 disabled	CI ZigBee Cluster ID	11
EO Encryption Options	0	I/O Setting	
KY Encryption Key	( )	D0 – AD0/DIO0 Configuration 1= Commissioning button	1
Serial Interfacing		D1 – AD1/DIO1 Configuration 0=disabled	0
BD Baud Rate	3 9600	D2 – AD0/DIO2 Configuration	0
NB Parity	0	D3 – AD0/DIO3 Configuration	0
SB Stop Bits	1	D4 – AD0/DIO4 Configuration	0
RO Packetization Timeout	3	D5 – DIO5/Assoc Configuration 1=Associated indicator	1
D7 – DI07 Configuration CTS Flow Control	0	P0 – DIO10/PWM0 Configuration	1
D6 – DI06 Configuration	0 disable	P1 – DIO11 Configuration	0
AT Command Options		P2 – DIO12 Configuration	0
CT AT command Mode Timeout	64	PR Pull up Resistor Enable	1FFF
GT Guard Times	3E8	LT Associate LED Blink Time	0
CC Command Sequence Character	2B	RP RSSI PWM Timer	28
Sleep Modes		I/O Sampling	
SM Sleep Mode No sleep (router)	0	IR IO Sampling Rate	0
SN Number of cyclic sleep periods	1	IC IC Digital IO Change Detection	0
SO Sleep Options	0	V+ Supply Voltage High Threshold	0
SP Cyclic Sleep Period (Valeur origien =20)	20	Diagnostic Commands	
ST Time before Sleep	1388	VR (2264) – Firmware Version	
PO Poll Rate	0	HV (1A44) – Hardware Version	
		AI (0) – Association Indication	
		DB (0) – RSSI of Last Packet	
		%V (0) – Supply Voltage	

Annexe 1.3 : Configuration "end device

Modules			
ZigBee end device at			
Networking		**Serial Interfacing**	
Id – Pan id	1	BD Baud Rate	3 9600
SC – Scan channels	1FFE	NB Parity	0
SD– Scan Duration	3	SB Stop Bits	1
ZS - ZigBee Stack profile	0	RO Packetization Timeout	3
NJ – Node Join Time	FF	D7 – DI07 Configuration CTS Flow Control	0
NW – Network Watchdog Timeout	0	D6 – DI06 Configuration	0 disable

JV – Channel Verification	**0** **disabled**	**AT Command Options**	
JN – Join Notification	0 disabled	CT AT command Mode Timeout	64
OP Operating PAN ID (FFD25DFC279EC686)	-	GT Guard Times	3E8
OI Operating 16 bit PAN ID (48C1)	-	CC Command Sequence Character	2B
CH Operating Channel (14)	-	**Sleep Modes**	
NC Number of Remaining Children (C)	-	SM Sleep Mode Cycle sleep)	4
Addressing		ST Time before Sleep	1388
SH Serial Number High (13A200)		SP Cyclic Sleep Period	20
SL Serial Number Low (<N° unique>)		SN Number of cyclic sleep periods	
MY 16 bit Network Address(5A9C)		SO Sleep Options	1
DH Destination Address High	0	PO Poll Rate	0
DL Destination Address Low	0	**I/O Setting**	
NI Node Identifier	Module x	D0 – AD0/DIO0 Configuration 1= Commissioning button	1
NH Maximum Hops	1E	D1 – AD1/DIO1 Configuration 0=disabled	0
BH Broadcast Radius	0	D2 – AD0/DIO2 Configuration	0
AR Many to One Route Broadcast Time	FF	D3 – AD0/DIO3 Configuration	0
DD Device Type Identifier	30000	D4 – AD0/DIO4 Configuration	0
NT Node Discovery Backoff	3C	D5 – DIO5/Assoc Configuration 1=Associated indicator	1
NO Node Discovery Options	0	P0 – DIO10/PWM0 Configuration	1
NP Maximum Number of Transmission Bytes (54)		P1 – DIO11 Configuration	0
ZigBee Addressing		P2 – DIO12 Configuration	0
SE ZigBee Source Endpoint	E8	PR Pull up Resistor Enable	1FFF
DE ZigBee Destination Endpoint	E8	LT Associate LED Blink Time	0
CI ZigBee Cluster ID	11	RP RSSI PWM Timer	28
RF Interfacing		**I/O Sampling**	
PL Power Level	4 Highest	IR IO Sampling Rate	0
PM Power Mode (1= boost mode enabled)	1 boost	IC IC Digital IO Change Detection	0
Security		V+ Supply Voltage High Threshold	0
EE Encryption enable	0 disabled	**Diagnostic Commands**	
EO Encryption Options	0	VR (2264) – Firmware Version	
KY Encryption Key	( )	HV (1941) – Hardware Version	
		AI (0) – Association Indication	
		DB (0) – RSSI of Last Packet	
		%V (B08) – Supply Voltage	

	VR	MY	ID	SH	SL	DH/DL
Module1	2264	DEC	0	13A200	4031E5DE	0/0
Module2	2264	5A9C	0	13A200	4031E5EB	0/0
Module3	2264	7452	0	13A200	4031E5F3	0/0
Module4	2264	A0B7	0	13A200	4031E5F4	0/0
Module5	2264	4729	0	13A200	4031E63A	0/0
Module A fil. Ext.	2064	0	0	13A200	403AF028	0/0
Module XBP24	2264	FFFE	0	13A200	403D136A	0/0

BIBLIOGRAPHIE

[AMIGO 2008], Projet Amigo Challenge - Site : http : //www.extra.research.philips.com/amigochallenge/index.htm

[BEAN 1997], Oracle/Sun Microsystems. "Java Beans 1.01 Specification". 1997. lien téléchargement : http : //java.sun.com/javase/technologies/desktop/javabeans/docs/spec.html

[Bieber], **Guy Bieber, Jeff Carpenter**, Introduction to Service-Oriented Programming, Septembre 2001 – Lien : http : //www.openwings.org/download/specs/ServiceOrientedIntroduction.pdf

[Box 1998], **D. Box**. "Essential COM". Addison-Wesley, 1998

[CAMIDO 2006], **N. BELHANAFI BEHLOULI, C. TACONET, G. BERNARD**, "An architecture for supporting Development and Execution of Context-Aware Component Applications". – Lien : http : //www-public.int-evry.fr/~taconet/LibreService/Publis/ICPS06Nabiha.pdf

[CARISMA 2003], **Licia CAPRA, Wolfgang EMMERICH, Cecilia MASCOLO**, "CARISMA, IEEE Transactions on Software Engineering, 2003. – Lien : http : //www.cs.ucl.ac.uk/staff/l.capra/publications/cem03.pdf

[CARMEN], **Paolo Bellavista, Antonio Corradi, Rebecca Montanari, Cesare Stefanelli**, Context-Aware Middleware for Resource Management in the Wireless Internet. - http : //zeus.ws.dei.polimi.it/is-manet/Documenti/pap-deis-2.pdf

[CCM 2006], OMG, "CORBA Component Model – V4.0 3.0". Avril 2006. Lien URL : http : //www.omg.org/technology/documents/formal/components.htm

[Cervantes 2004], **H. Cervantes and R. S. Hall**, "Autonomous Adaptation to Dynamic Availability Using a Service-Oriented Component Model," in *International Conference on Software Engineering (ICSE)*, Edinburgh, 2004.

[Chalon 2004], **René Chalon.** Réalité Mixte et Travail Collaboratif : IRVO, un modèle de l'Interaction Homme –Machine., Thèse de doctorat, Ecole Centrale de Lyon, 2004.

[CORBA 2008], OMG Released Versions Of CORBA, Version 3.1, Janvier 2008 - Lien : http : //www.omg.org/spec/CORBA/

[CORTEX 2003], Projet CORTEX, Juin 2003 - Site : http : //cortex.di.fc.ul.pt/index.htm

[Delotte 2006], **Olivier Delotte.** CoCSys : une approche basée sur la construction d'un modèle comportemental pour la conception de systèmes collaboratifs mobiles. Thèse de doctorat, Ecole Centrale de Lyon, 2006.

[DIGI 2010], Utilitaire graphique pour configurer module XBee- Lien : http : //www.digi.com/support/supporttype.jsp?sfid=0&pgid=12&fr=Y&tp=0 , Mai 2010

[DPWS, OASIS 2009], Devices Profile for Web Services , Janvier 2009 – Lien : http : //docs.oasis-open.org/ws-dd/ns/dpws/2008/09

[EJB 2001], **R. Monson-Haefel**. "Enterprise JavaBeans". O'Reilly & Associates, 3rd edition, octobre 2001.

[Fractal 2004], The Fractal Component Model. Février 2004. Lien : http : //fractal.ow2.org/specification/index.html

[GAIA 2002], **Manuel Román, Christopher Hess, Renato Cerqueira, Anand Ranganat, Roy H. Campbell, Klara Nahrstedt** - Gaia : A Middleware Infrastructure to Enable Active Spaces, Janvier 2002 - Lien : http : //gaia.cs.uiuc.edu/papers/GaiaSubmitted3.pdf

[GAIA, TAO 1999], **Manuel Roman, Fabio Kon, and Roy H. Campbell**. Design and Implementation of Runtime Reflection in Communication Middleware : the dynamicTAO Case. In Workshop on Middleware, ICDCS'99, May 1999.

[iPOJO], Projet felix Apache Mai 2010 - Site : http : //felix.apache.org/site/apache-felix-ipojo.html

[Johnson 1993], **Ralph Johnson**. How to design frameworks. Notes for OOPSLA 93, October 1993.

[Johnson 1997], **Ralph E. Johnson**. Components, frameworks, patterns. Dans ACM SIGSOFT Symposium on Software Reusability, pages 10-17, 1997

[Jini River], Projet River (Jini), Mars 2010 - Site : http : //incubator.apache.org/river/RIVER/index.html

[Krakowiak 2003], **Sacha Krakowiak.** Middleware Architecture with Patterns and Frameworks- Lien : http : //sardes.inrialpes.fr/~krakowia/MW-Book/

[MDA 2001], OMG, Model Driven Architecture, MDA Guide V1.0.1 Juin 2001, Site : http : //www.omg.org/cgi-bin/doc?omg/03-06-01

[Networking Toolkit 2009], – Projet Open source écrit en C# créer par Michael Schwarz – Lien : http : //mftoolkit.codeplex.com/

[OASIS, SCA 2007], Normalisations – Spécification de SCA par OASIS, Aout 2007 : http : //www.oasis-opencsa.org/sca

[OSGi 2009], "OSGi Service Platform Release 4", Version 4.2, Septembre 2009.
Lien URL : http : //www.osgi.org/Specifications/HomePage

[Oxygen], Projet Oxygen du MIT : Pervasive Human-Centered Computing – Site : http : //oxygen.csail.mit.edu/Overview.html

[Papazoglou 2003], **M. P. Papazoglou, D. Georgakopoulos**. Service-Oriented Computing. Communications of the ACM, October 2003

[Pujolle 2008] **Guy Pujolle**. Les réseaux Edition 2008. Edition Eyrolles, septembre 2007

[Pujolle 3Edit] **Guy Pujolle**. Cours réseaux et télécoms. Edition Eyrolles; octobre 2008

[RCSM 2004], Projet RCSM du département d'informatique de l'université de L'Arizona – mise à jour avril 2004 : - Site : http : //dpse.asu.edu/rcsm/

[SAFRAN 2005], **Pierre Charles DAVID** : Développement de composants Fractal adaptatifs : un langage dédié à l'aspect d'adaptation.- Thèse de doctorat, Université de Nantes UFR Sciences et

Technique, Juillet 2005 - Lien : http : //pagesperso-orange.fr/pcdavid/research/papers/2005/phd/david_phd2005.pdf

[SCA, OSOA 2010], Open Service Oriented Architecture - projet SCA mise à jour mai 2010- Lien : http : //www.osoa.org/display/Main/Home

[Service Web 2004], Web Services Architecture Requirements, février 2004
http : //www.w3.org/TR/wsa-reqs/

[SLP 1997], Protocol de découverte de services de services développé par IETF SvrLoc.- RFC 2165, juin 1997 – lien : http : //www.ietf.org/rfc/rfc2165.txt

[SOCAM], **Tao Gu, Hung Keng Pung, Da Qing. Zhang** - A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications* –Lien : http : //lucan.ddns.comp.nus.edu.sg : 8080/PublicNSS/Publications/JNCA_1_gutao.pdf

[Szyperski 1996], Clemens Szyperski. Independently extensible systems – software engineering potential and challenges. Dans *Proceedings of the 19th Australian Computer Science Conference*, Melbourne, Australie, 1996.

[Szyperski 2002], **Clemens Szyperski** . -- *Component Software : Beyond Object-Oriented., ACM Press, Component Software Series, A*ddison-Wesley , 2002.

[UPnP 2010], Universal Plug and Play Forum - Lien : http : //www.upnp.org/ , Mai 2010

[WComp 2009a], .**Daniel Cheung-Foo-Wo** - Adaptation dynamique par tissage d'aspects d'assemblage – Thèse de doctorat, de l'université de Nice, 2009.

[WComp 2009b],. **Jean-Yves Tigli, Stéphane Lavirotte, Gaëtan Rey, Vincent Hourdin et Michel Riveill** - Lightweight Service Oriented Architecture for Pervasive Computing. International Journal of Computer Science Issues (IJCSI), volume 4, numéro 1, pages 1-9, septembre 2009. ISSN 1694-0814

[Weiser 1991], **M. Weiser** – The Computer for the 21 st Century, Scientific American, vol 265, N° 3, sept. 1991, pp. 66-75

[Wellner 1993], **P. Wellner, W. Mackay, R. Gold** – Computer Augmented Environments : Back to the Real Word. Special Issued of Communications of the ACM, vol. 36, N°7, July 1993

Etat de l'art, proposition et mise en oeuvre d'un intergiciel pour la communication dans un environnement réel augmenté entre acteurs mobiles

Mémoire d'Ingénieur CNAM, Lyon 2010

RESUME

Ce rapport présente l'étude, la conception et la mise en œuvre d'un intergiciel supportant les applications prenant en compte les acteurs mobiles évoluant dans un environnement réel augmenté. Dans cet environnement est réparti un ensemble de capteurs sans fil, apportant des informations contextualisées aux applications et aux acteurs. La première partie traite de la technologie Zigbee qui est spécifiquement conçue pour des capteurs sans fil disséminés, car peu gourmands en énergie et donc particulièrement autonomes. L'inconvénient des technologies sans fil est la fragilité de la liaison radio qui impacte la qualité de service, ce qui doit être pris en compte par l'intergiciel. La deuxième partie présente la plateforme WComp choisie comme réponse à la problématique pour mettre en œuvre un prototype. Enfin, deux études de cas décrivent deux mises en œuvre de notre prototype.

SUMMARY

This report presents the study, the design and the implementation of a middleware supporting applications taking into account mobile actors moving in an augmented real environment. Wireless sensors are disseminated in this environment which bring contextualised informations to the applications and to the actors. The first part presents Zigbee technology which is specificaly designed for wireless sensors networks, using few energy and therefore particulary autonomous. The disadvantage of wireless technologies is the fragility of the radio transmission, impacting the quality of service, which must be taken into account by the middleware. In the second part, we present the WComp platform chosen to implement a prototype. Finally, two case studies describe two implementations of our prototype.