Numéro d'ordre : **Année 2012**

Université Lille 1 - Sciences et Technologies

THESE

présentée et soutenue publiquement le 20 Décembre 2012

En vue de l'obtention du grade de

DOCTEUR DE L'UNIVERSITÉ LILLE I

Discipline : Informatique

par

Benjamin Barbry

Usine logicielle pour la conception d'environnements pervasifs : Application au e-Retail

Composition du jury

Président :	Pr Gilles Grimaud	LIFL, Université Lille 1
Rapporteurs :	Dr Thibault Carron, HDR	Université de Savoie
	Pr Bertrand David	École Centrale de Lyon
Examinateur :	Pr Amel Bouzeghoub	Télécom SudParis
Directeur de thèse :	Pr Alain Derycke	LIFL, Université Lille 1
Co-directeur de thèse :	Dr Thomas Vantroys	LIFL, Université Lille 1

Laboratoire d'Informatique Fondamentale de Lille - Équipe NOCE

Remerciements

Je tiens à remercier *Thibault Carron* et *Bertrand David* d'avoir accepté d'être rapporteurs de cette thèse et de m'avoir apporté leurs conseils sur mon travail. Je tiens aussi à remercier *Gilles Grimaud* pour m'avoir fait l'honneur d'accepter la présidence du jury de thèse. Je remercie également *Amel Bouzeghoub* d'avoir accepté d'être membre du jury.

Je remercie *Alain Derycke*, mon directeur de thèse, pour m'avoir soutenu, aidé et conseillé. J'ai particulièrement apprécié son enthousiasme et ses idées.

J'ai effectué mon travail au sein de l'équipe NOCE et je tiens à remercier les membres de l'équipe auprès de qui j'ai beaucoup appris. Oui, c'est bien à vous, *Yvan Peter* et *Eric Lepretre*, à qui je pense. Merci pour vos conseils avisés, votre bonne humeur et vos "jeux de mots".

Un merci particulier pour "mon binôme" *Philippe Laporte* sans qui tous ces travaux n'auraient pas aboutis. Merci pour ta bonne humeur, ton accueil au sein de NOCE et surtout merci de m'avoir supporté si longtemps.

Mais que serait une thèse sans un bon encadrant. Je tiens à remercier *Thomas Vantroys* qui a eu la patience de tenir ce rôle jusqu'au bout. Ce serait trop long de tout expliquer, nos expéditions, nos jeux,..., alors je te remercie simplement pour tout.

Mes travaux au sein du projet p-LearNet m'ont permis de rencontrer des personnes d'horizons différents, qui ont contribué à m'ouvrir l'esprit et qui m'ont fait partager leurs richesses. Je remercie en particuliers *Sylvie Lerouge* pour son enthousiasme à l'innovation, *Serge Garlatti* pour sa bonne humeur et son accueil chaleureux lors de nos expéditions à Brest, et enfin *Cuong Pham Nguyen* pour nos échanges culturels.

Grâce à l'école doctorale, j'ai eu l'occasion de rencontrer d'autres personnes dans la galère [1] avec lesquels j'ai noué une amitié forte. Merci à *Matthieu* et *Sylvain* pour ses bons moments sur et hors scène. Merci surtout à *Tony* et *Asli*, docteurs en informatique à l'heure où je vous écris, et dont les rires hanteront à jamais le M3.

Merci également à tous mes amis, *Tonio, Hochet, Ch'doud, Lolo, Jon, Cheb, Doro* et tant d'autres, pour tous les moments passés ensemble et qui m'ont permis d'évacuer la pression, tout en en vidant quelques unes quand cela devenait nécessaire.

Je remercie ma famille pour son soutien et sa confiance. En particulier mes parents qui m'ont permis de réussir mes études et qui depuis toujours m'ont accordé leur soutien et leur confiance.

Finalement, j'entends souvent dire qu'une thèse, ça change la vie. Et que de changements depuis le début. Le petit *Marceau* a d'abord fait de moi un parrain heureux. Ensuite *ma bidule* est devenue ma femme. Merci *Sophie* notamment pour ta patience, ta compréhension et ton soutien...que de choses réalisées en deux semaines. Et enfin, une dernière pensée à *Capucine* qui m'a transformé en papa comblé. Je suis désolé, mais j'ai maintenant du temps libre pour m'occuper de vous.

Cette liste n'est pas exhaustive. Que ceux qui auraient été oubliés ne m'en tiennent pas rigueur.

1. des thésards donc

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

The most profound technologies are those that disappear.
They weaves themselves into the fabric of everyday life
until they are indistinguishable from it.
Mark Weiser, [WGB99]

L'objectif de l'informatique ubiquitaire (*ubiquitous computing*) tel que défini par Mark Weiser il y a près de vingt ans, "*Ubiquitous computing has as its goal the nonintrusive availability of computers throughout the physical environment, virtually, if not effectively, invisible to the user*" [Wei93a] , n'est toujours pas devenu une réalité prenant place dans la vie du plus grand nombre de personnes. La réduction de la taille et l'évolution des formats des équipements (de la station de travail à l'ordinateur portable au *smartphones* et à la tablette) et des modes d'interaction n'ont pas fait disparaître l'objet technique en lui-même, *"the computer nonetheless remains largely in a world of its own"* [Wei99] . Bien que communiquant de plus en plus facilement notamment via des liaisons sans-fil permanentes (WiFi, bluetooth, 3G, ...), le terminal utilisateur reste dans sa bulle, sans jamais réellement interagir avec l'environnement de l'utilisateur (au delà de la simple fourniture de services géolocalisés). Nos équipements sont du 21^{eme} siècle mais notre environnement direct (bâtiment, mobilier, ...) du 20^{eme} siècle. Or, "*The real power of concept comes from any of these devices - it emerges from the interaction of all of them*" [Wei99].

Ce rapport à l'environnement passe également par de nouveaux outils et de nouveaux mécanismes d'interaction. L'ordinateur, et le couple clavier/souris, est un outil très générique [Kum09], il permet de réaliser un ensemble de tâches très varié, allant du traitement de texte aux jeux vidéos, en passant par la CFAO, le traitement d'image, ... Or, les outils que nous "oublions" sont ceux qui sont spécialisés. Si nous voulons faire "disparaître" l'ordinateur en tant qu'objet pour qu'il devienne enfin une multitude d'outils aux formes et aux fonctionnalités variées mais spécialisés et répondant chacun à des tâches

humaines spécifiques, il est nécessaire de faire communiquer ces outils et de coordonner leurs différentes actions.

Le dernier point, selon nous, pour la création d'environnement ubiquitaire réside dans la capacité et surtout la facilité de créer de nouveaux outils/artéfacts en combinant des services fonctionnels et des services techniques (à la fois matériel et logiciel) afin de pouvoir les produire et les tester rapidement, "*A key part of our design philosophy is to put devices in everyday use*" [Wei93b]. Il faut pour cela offrir des outils de composition rapide permettant de s'abstraire des protocoles et des modes de communications sous-jacent. Nous nous inspirons également de l'approche *bootstrapping* définie par Douglas Engelbart [Eng68, Eng62]. Il préconise que celui qui travaille à l'élaboration et/ou au déploiement de nouveaux outils pour stimuler l'intelligence collective devrait utiliser ce qu'il construit pour améliorer sa propre efficacité. Le concepteur devient ainsi utilisateur de sa propre technologie, ce qui facilite l'évolution et la collaboration avec les utilisateurs finaux.

Notre thèse s'intéresse donc au prototypage rapide d'application ubiquitaire et plus particulièrement dans le domaine du commerce ambiant [DCV07]. Deux types d'applications sont visés, rendre l'environnement porteur d'informations et faciliter la formation des vendeurs directement sur la surface de vente [DVBL08]. Nos travaux furent réalisés, au sein de l'équipe NOCE du LIFL, dans le cadre du projet ANR p-LearNet et du groupe de travail **E**nvironnements de **V**ente **I**ntelligents et **R**éalité **A**ugmentée (EVIRA) du **P**ôle de compétitivité des **I**ndustries du **COM**merce (PICOM).

1.1 Le projet p-LearNet

1.1.1 Présentation générale

Financé par l'Agence Nationale de la Recherche, p-LearNet est un projet sur 3 ans dont le but est d'explorer l'apport des TICs pour un champ d'application large : l'apprentissage humain n'importe où, n'importe quand, dans n'importe quel contexte technologique et organisationnel. Nous avons travaillé avec trois partenaires fournissant des terrains d'expérimentations réels : Auchan, La Poste et l'UMVF [2]. Pour ces partenaires très différents d'un point de vue organisationnel, le processus d'apprentissage est une plus-value non seulement pour les personnes qui y travaillent, pour les rendre plus efficaces et réactifs, mais également pour les usagers. L'avantage d'avoir des terrains d'expérimentation si distincts est de nous permettre de tester différentes solutions dans des contextes spécifiques, ce qui nous permettra ainsi de définir les invariants pertinents pour les situations d'apprentissage en mobilité.

1.1.1.1 Problématiques

Le projet p-LearNet veut prouver la faisabilité d'un nouveau domaine de fonctionnalités pour un domaine émergeant appelé "Apprentissage pervasif"(ou p-Learning) [Der08] qui est plus qu'une simple intégration des TICs et des réseaux de communication. Le p-learning vient du constat suivant : on apprend tout le temps, souvent de manière informelle, l'apprentissage, pouvant être considéré comme

2. **U**niversité **M**édicale **V**irtuelle **F**rancophone

des échanges et des constructions de connaissances formelles et informelles, est crucial pour le développement [STV05]. Un système informatique conçu pour supporter l'apprentissage doit donc être capable de s'adapter à toute forme d'apprentissage. Ce projet est alors une réflexion sur la conception d'une façon innovante d'apprendre dans un contexte d'intelligence ambiante. Cela se traduit par la prise en compte non seulement de l'ubiquité des systèmes d'apprentissage et de la mobilité des apprenants, mais aussi des nouvelles propriétés requises pour ce nouveau mode d'apprentissage (supporter l'activité professionnelle, être capable de reprendre une ancienne activité d'apprentissage interrompue par un besoin métier ...). De plus, le potentiel des interfaces utilisateurs multimédia, multicanales et multimodales, sensibles à la personnalité de l'utilisateur/apprenant, est également à considérer [Che06]. Derrière les termes *multicanal* et *multimodal*, nous entendons un choix riche de combinaisons de différents terminaux utilisateur (ordinateur, smartphone, ...) avec différents types de réseaux (topologie, qualité de service, protocoles, coûts...), en incluant la voix, la vidéo et la communication pair-à-pair. Les systèmes d'apprentissages pervasifs impliquent aussi la capacité, dans un champ particulier d'application, de maîtriser les processus de découvertes, de composition et d'adaptation de services contextuels et d'interactions avec les utilisateurs. Ces sytèmes permettent également de surmonter le problème de complexité provenant des nombreuses possibilités de combinaisons des différents éléments.

L'originalité du projet est de conduire une étude exploratoire de l'informatique mobile vue comme une extension des possibilités offerte à une personne par son environnement, avec une étude sur les approches "services et usages" provenant de l'intelligence ambiante. Ces études ont principalement été conduites séparément auparavant. Les études proposées dans nos travaux sont enracinées dans un champ stratégique d'application, l'apprentissage, donnant un ancrage dans des usages réels et des besoins d'entreprises et d'organisations. Ces entreprises et organisations sont représentées dans le consortium p-LearNet sous forme de sites pilotes, couvrant ainsi un large panel de besoins. Il apparait donc évident que les problèmes génériques comme la prise en compte du contexte ou l'adaptation dynamique d'un système ne peuvent être résolus dans une approche applicative indépendante.

Un autre problème clé est le besoin d'un suivi continu de l'utilisateur au sein d'une grande variété de contextes, et notre effort sera donc porté dans cette direction selon deux niveaux. Le premier se situe au sein de l'infrastructure d'intermédiation, entre le monde des services et celui de la communication, en ajoutant plus d'intelligence pour fournir des services sans interruption, quels que soient les canaux utilisés. Le second niveau se place en dehors du système, pendant les expérimentations, en fournissant des instrumentations spécifiques des plate-formes d'expérimentations, en prenant en compte la nature distribuée et changeante des interactions et des problèmes de l'évaluation de l'usage à distance.

Certains résultats du projet sont utiles dans d'autres domaines, comme le commerce ubiquitaire (combinaison dynamique du e-commerce, mobile commerce et des lieux de vente classiques augmentés par les technologies). Nous pouvons également transposer nos travaux à d'autres domaines, car les thèmes abordés peuvent s'appliquer au-delà du périmètre du projet : l'intégration des technologies et la production d'infrastructures adaptatives ; la production de services orientés usages pour les utilisateurs/clients ; l'usage, l'efficacité et l'acceptation de ces environnements de nouvelles technologies par un grand nombre d'utilisateurs potentiels.

1.1.2 Démarche

L'usage est un élément clé de ce projet et nous proposons de définir notre projet comme étant centré sur les usages et la co-évolution. L'idée est d'organiser notre travail en petit cycles d'études d'usages et de conception de prototypes, ce qui nous permet d'appliquer certains des concepts de l'*eXtrem programming* [Bec02]. L'*eXtrem programming* est une méthode de gestion de projet informatique dite *agile*. Elle se base, en les poussant à l'extrême, sur des principes de développements du génie logiciel notamment l'intégration de l'utilisateur final dans les équipes de développement, des cycles de développement courts afin de tester régulièrement sur les terrains les évolutions du logiciel pour s'adapter en permanence aux réels besoins. Cela s'accompagne de *refactoring* régulier du code qui est produit. En suivant ces principes, les fonctionnalités des prototypes sont incrémentales (un simple service avec un seul média et un seul canal pour commencer puis progressivement plusieurs services avec plusieurs média et en utilisant la multicanalité). Ces développements de petits prototypes permettent des réactions rapides et ainsi une émergence des besoins réels.

1.1.3 Terrain d'expérimentation de nos travaux de thèse

Dans le cadre du projet p-Learnet, nos travaux de thèse ont principalement été réalisés au sein du terrain d'expérimentation offert par Auchan. L'apprentissage dans ce type de structures concerne une multitude de personnes de profils différents. La conception et l'ingénierie des systèmes d'apprentissage pervasifs doivent être considérés comme des problèmes interdisciplinaires nécessitant l'intégration de différentes approches scientifiques issues de l'informatique, des sciences de l'éducation, du commerce, des sciences sociales ...Les travaux portant sur l'apprentissage dans ce cadre se concentre principalement sur la manière de supporter les méthodes d'apprentissage individuel ou de groupe à travers des principes pédagogiques et comment enrichir la connaissance de l'apprenant. Les partenaires identifient directement sur le terrain les problèmes et les besoins portant sur la qualité et l'efficacité des informations et des services afin d'accroitre la part de marché et les buts d'apprentissage correspondant. Plusieurs scénarios (voir annexe B) ont été mis en place selon 2 situations de travail et d'apprentissage avec un vendeur et un client en tant qu'apprenant :

1. le vendeur ou le client en dehors des limites du magasin : le vendeur en réserve, ou le client chez lui ;
2. le vendeur dans son rayon, seul ou avec un client disposant de ressources d'espaces intelligents (grands écrans LCD, imprimantes, RFID, ...) qui les entourent [DCV07].

1.1.3.1 Résultats obtenus

À partir de nos observations de terrain, nous pouvons remarquer que, dans le cadre du commerce, le contexte de travail est constamment mis à jour pendant la vente : un vendeur peut communiquer avec des clients pendant qu'il révise ses connaissances, vérifie son inventaire ou contacte des fournisseurs. Il apparait qu'une part substantielle de l'apprentissage ne se passe pas pendant les temps de formation mais pendant l'activité de travail. Les supports d'apprentissage et de travail doivent donc être intégrés.

Un système d'apprentissage doit surmonter trois principaux obstacles : la contrainte temporelle, les supports d'apprentissage inadaptés dans un contexte de travail, et la déconnexion, cognitive et structurelle, entre le travail, la connaissance et l'apprentissage [Far07]. Nous avons conçu dans ce cadre un système d'aide à l'apprentissage et aux activités de travail qui se présente sous la forme d'un appareil mobile que nous avons nommé Personal Selling Assistant (PSA). Pour cela, nous avons notamment développé un intergiciel d'intermédiation que nous présenterons en détail dans le chapitre 5. Dans un des scénarios, par exemple, un vendeur équipé de cet outil mobile, l'aidant dans ses différentes tâches, pourrait profiter de l'absence de clients dans son rayon pour réviser ses connaissances sur les produits et les techniques de vente ou reprendre une précédente activité d'apprentissage pour améliorer ses connaissances, ou encore il/elle pourrait vérifier les étiquettes des produits ou accéder à des informations relatives aux produits. Pendant une vente, le vendeur peut également s'aider de son périphérique mobile comme un conseiller qui lui délivre des conseils de vente en fonction de la situation. Mais un appareil seul, même relié à un réseau de type Internet, ne pourrait répondre à nos besoins. Comme souvent dans les systèmes pervasifs, l'environnement dans lequel évolue l'utilisateur doit être intégré au système et devenir "intelligent". D'un point de vue technique, nous définissons alors le lieu de travail du vendeur comme étant un Espace de Vente Intelligent (EVI) . Un EVI tire profit du développement des TICs et de l'évolution du monde du commerce.

1.2 EVIRA : Évolution du commerce et le rôle des TICS

Le travail que nous avons réalisé sur la notion d'environnements intelligents a servi de base à la création du groupe de travail EVIRA [3] du PICOM. Ce groupe de travail vise à réfléchir sur les infrastructures pour le commerce situé de demain.

L'ensemble de ces travaux rentrent dans le contexte du e-retail. Le e-retail concerne l'utilisation des nouvelles technologies de communication et d'information pour le commerce. Les principales innovations liées au e-retail traitent principalement de l'augmentation des lieux de vente par les TIC. Nous pouvons par exemple citer les armoires intelligentes [WW06], les « caddies augmentés » « smart carts »[Kou03], les affichages publicitaires intelligents. Le e-retail n'est pas juste issu de l'évolution technique et technologique. Il est dans la continuité de l'évolution générale du commerce.

1.2.1 Évolution du commerce

Le monde du commerce a subi de profonds changements. Ces changements proviennent notamment de l'impact des nouvelles technologies dans la vie de tous les jours. En effet, la popularisation des TIC [4], notamment l'accès facile à Internet, a profondément modifié les habitudes des "personnes". Les acteurs du marché ont du s'adapter aux attentes des clients réels et potentiels.

Ces technologies, comme l'Internet et les technologies des objets mobiles communicants, ont déjà bouleversé le monde du commerce, avec le développement important du commerce électronique, qu'il

3. **E**nvironnements de **V**ente **I**ntelligents et **R**éalité **A**ugmentée
4. **T**echnologies de l'**I**nformation et de la **C**ommunication

soit un prolongement des approches du Marketing Direct ou l'extension des capacités du commerce avec espaces de ventes traditionnels. Nul doute que l'avènement du Web 2.0 va encore amplifier ce développement. À première vue cependant, il apparaît que les lieux de vente physiques, de la boutique à l'hypermarché en passant par les autres formats commerciaux, n'ont pas subi la même mutation sous l'impact des nouvelles technologies. C'est particulièrement vrai pour tout ce qui touche à la relation directe avec le client. Mais cela l'est moins pour ce qui concerne bien sûr, la gestion commerciale de l'espace de vente avec l'informatisation de la gestion des stocks et approvisionnements et des fonctions de caisses.

Néanmoins si l'on est observateur au sein des espaces de ventes actuels on voit quelques applications locales des nouvelles technologies : comme la possibilité pour le client de lire le code-barres d'un produit pour en connaître le prix, lire une bande annonce d'un DVD ou des extraits sonores d'un CD, choisir des cartouches d'encre pour imprimantes ou des essuie-glaces sur un écran tactile, voire des surfaces d'affichage électroniques pour les promotions (mais non interactives, ni personnalisées). Il faut constater que la démarche de conception de ces nouveaux espaces de ventes n'a pas encore intégré de manière systématique tous les apports potentiels des nouvelles technologies.

Du point de vue des systèmes d'information interactifs, il est apparu récemment de nouvelles modalités, comme la Réalité Augmentée qui enrichit les espaces et les objets réels et physiques, à partir d'éléments d'informations produits dynamiquement et en interaction avec le client par un système informatique. Parmi les exemples qui commencent à poindre sous l'impulsion de certains grands fournisseurs de système de gestion des espaces de ventes on pourrait mettre en avant des vitrines, des étagères, des cabines d'essayage, plus interactives et utilisant souvent le potentiel des étiquettes sans contact de type RFID [5] ou NFC [6] [NFC] pour la gestion des produits ainsi mis en scène.

1.2.2 Le groupe de travail EVIRA

Ce projet provient du constat que l'on peut tirer de la partie précédente : les projets d'innovation autour du commerce, et notamment sur les lieux de vente, sont souvent réalisés sous forme de prototypes ou alors réalisés pour une utilisation bien spécifique et non modifiable (changement de magasin ou de produits impossibles). Une étude de ces projets nous a permis de dresser les remarques suivantes :

- il manque un cadre de conception dédié à l'innovation
- il n'existe pas de mutualisation concernant tous ces projets
- il n'y pas de réelle recherche concernant l'usage, il s'agit principalement de Push technologique, fait par les fournisseurs de technologies TIC pour le commerce
- il n'y pas toujours de réelles interactions avec le client, les systèmes proposés sont principalement cantonnés dans la diffusion passive d'informations

Ce projet vise donc une réflexion globale afin de proposer des standards pour une production industrielle de projets innovants à destination du commerce. Nous suivons une démarche de prospection. Dans le cadre de nos travaux de thèse, en nous basant à la fois sur les observations terrain réalisées dans

5. **R**adio **F**requency **ID**entification
6. **N**ear **F**ield **C**ommunication

le cadre du projet p-Learnet et sur les différents projets décrits dans la littérature, nous avons extrait les invariants des différents espaces de ventes et des équipements qui les composent. Ces invariants nous permettent de proposer une modélisation d'un espace de vente complet ou d'un sous-ensemble. La modélisation proposée concerne toutes les facettes de l'innovation sur les lieux de vente : infrastructure, interaction Homme machine, usages, services proposés, ...Pour le déploiement sur le terrain, nous nous appuyons sur l'infrastructure que nous avons développée et mise en œuvre dans le cadre du projet p-Learnet. Elle permet notamment de prévoir et d'inclure les accès aux systèmes d'information des enseignes, les interactions possibles avec les utilisateurs, clients ou employés, une reconfiguration facile et rapide pour s'adapter aux différents environnements, et ce dans un souci d'industrialisation. Cette partie sera traitée en détail dans le chapitre 6.

1.3 Contribution de la thèse et plan

1.3.1 Démarche et méthodologie

Nos travaux de thèse visent à simplifier le développement d'applications ubiquitaires dans le domaine du commerce ambiant. Pour cela nous avons suivi une approche transversale, depuis les usages jusqu'à l'infrastructure logicielle en passant par toutes les étapes nous permettant d'outiller cette transversalité. Nous avons pour cela profité de la multidisciplinarité de l'équipe NOCE en nous appuyant sur les points forts de l'équipe en terme de compétences en Interaction Homme-Machine, en Génie Logiciel et sur le support aux activités d'apprentissage. Afin d'arriver à la fabrique logicielle qui sera détaillée dans les chapitres suivants, nous avons suivi une approche basée sur la co-conception [BLM02], l'étude des usages et une méthodologie de type *eXtrem Programming* [Bec02]. Cette démarche a été l'un des fondements du projet p-LearNet. Afin de proposer de nouveaux usages, il est nécessaire et obligatoire de mettre les utilisateurs au cœur du processus de conception et cela dès son amorçage [Org10].

Pour cela, nous avons suivi une approche ascendante telle que décrite dans la figure 1.1.

Afin d'amorcer le travail de co-conception, nous avons élaboré, dans le cadre du projet p-LearNet, un ensemble de scénarios de base. Ils servent à planter le décor et permettent ainsi de mieux se représenter le contexte, les différents acteurs, les différentes tâches, les contraintes et les problématiques inhérentes. Les différents scénarios se déroulent au sein d'une grande chaîne d'hypermarchés et se basent sur des situations proches de la réalité du terrain. Nous avons adopté les points de vue de trois acteurs principaux dans le domaine qui nous concerne : ceux d'un vendeur et d'un client, pour introduire le monde du commerce, et celui d'un créateur d'EVI pour introduire le rôle de l'usine logicielle. Ces trois scénarios sont présentés dans l'annexe B. Ils nous ont permis de réaliser un premier prototype d'outils de support à l'apprentissage et aux activités de travail, contenant un minimum de fonctionnalités mais permettant d'initier la co-conception avec les utilisateurs. Il est en effet plus facile de discuter sur des artefacts concrets offrant ainsi la possibilité de présenter les outils à venir. Cette première phase a permis de raffiner les scénarios et de décider des premières fonctionalités à implémenter. À partir de ces recommandations, une seconde version du prototype fut développée et déployée sur le terrain afin de vérifier les hypothèses initiales (i.e. intégration des activités d'apprentissage dans les activités

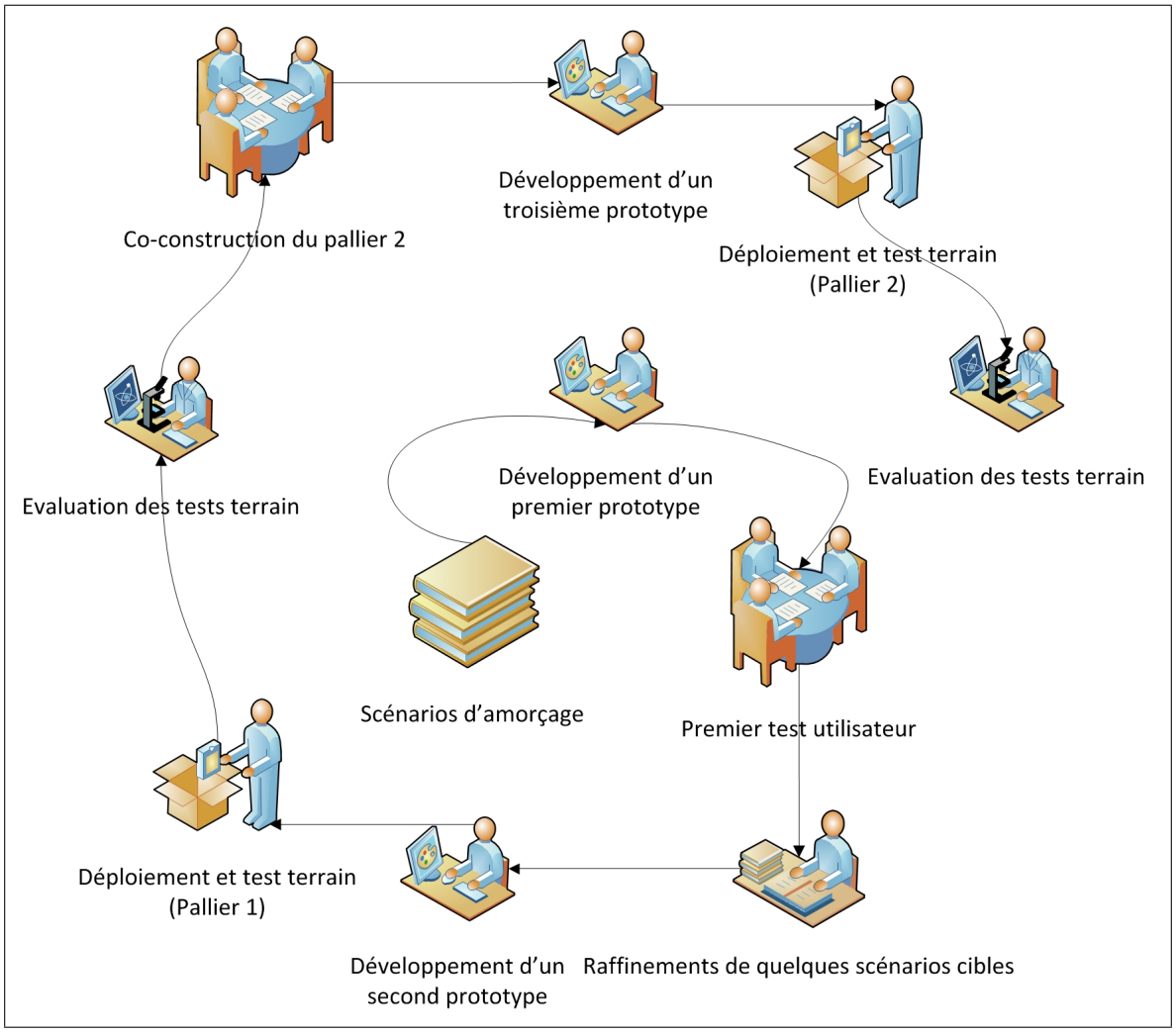

FIGURE 1.1 – Cycle de développement

de travail, utilisations des ressources offertes par l'environnement de travail, ...). Ce test grandeur nature a permis d'éprouver l'infrastructure logicielle de communication que nous avons développée et a permis de dégager de nouvelles fonctionnalités notamment liées à la communication et au partage de connaissance entre utilisateurs. Ce test nous a également permis de mieux appréhender les fonctions et les besoins pour la création d'environnement ubiquitaire pour le commerce.

À partir de l'ensemble de ces résultats, nous avons recommencé un cycle de co-conception, développement et tests sur le terrain. Nous avons ainsi pu acquérir une bonne connaissance du domaine et des besoins pour la conception d'applications ubiquitaires. En nous basant sur ces éléments, nous avons entamé la conception et le développement d'une fabrique logicielle spécialisée pour les environnements de vente ubiquitaires.

1.3.2 Notre contribution

Nos travaux de thèse portent sur la définition, la conception et le développement d'une fabrique logicielle permettant la conception, à plusieurs niveaux, d'environnements de vente intelligents.Comme le montre la **figure 1.2**, cette fabrique logicielle couvre les aspects :

- modélisation : nous avons décrit des modèles permettant de représenter des espaces de vente et les interactions possibles qu'ils proposent. Ces modèles permettent également de s'adapter en fonction des rôles des différents acteurs, que nous avons définis précédemment.
- conception : nous avons travaillé sur l'implémentation d'un outil de conception graphique d'espaces de vente intelligents, à partir des modèles.
- exécution : nous avons implémenté une infrastructure technique permettant d'exécuter les entités logicielles générées par le modeleur.

La solution proposée ne propose pas de lien entre les différentes parties mais offre un aperçu de la faisabilité d'une telle solution. Afin de tester nos différentes hypothèses concernant le commerce du futur, nous avons aussi implémenté un environnement de vente intelligent de test. Pour cela, nous avons pu, dans le cadre du projet de recherche p-LearNet, augmenter un rayon au sein d'un supermarché. Nous avons placé différents écrans diffusant de la publicité et avons fourni un appareil mobile aux vendeurs. Cet appareil leur sert à contrôler leur environnement (ici les écrans) mais il s'agit également d'un outil d'assistance à la vente, sur lequel nous reviendrons plus tard.

1.3.3 Plan du document

La suite de ce document se décompose en cinq chapitres.

Le **chapitre 2** dresse un état de l'art sur le commerce et l'utilisation des TIC dans les surfaces de ventes. Nous chercherons notamment à trouver les invariants des différents scénarios et des différents mécanismes mis en place.

Le **chapitre 3** permet de donner un aperçu des évolutions technologiques dans le domaine de l'intelligence ambiante. Ce chapitre nous éclaire sur les points importants à prendre en compte lors de la conception d'une application pervasive.

Le **chapitre 4** soulève les problématiques auxquelles nos travaux répondent. En plus des problématiques liées au commerce et à l'intelligence ambiante, nous mettrons en évidence le besoin d'industrialisation pour la conception d'espaces de vente intelligents. Ce chapitre permet de ppresenter l'usine logicielle comme une solution adéquate.

Le **chapitre 5** est le premier chapitre présentant nos travaux. Nous commençons par présenter le premier élément de notre usine logicielle, la couche de communication. La fin de ce chapitre est dédié à nos prototypes réalisés à partir de cette plate-forme.

Le **chapitre 6** présentera en détail la fabrique logicielle, de la modélisation à l'éxecution sur la plate-forme de communication, en passant par la partie génération de code. Nous présentons également en fin de chapitre notre prototype de vitrine intelligente réalisé à l'aide de notre usine logicielle.

Enfin, le **chapitre 7** termine ce document en résumant nos contributions, et en montrant les différentes perspectives possibles suite à nos travaux.

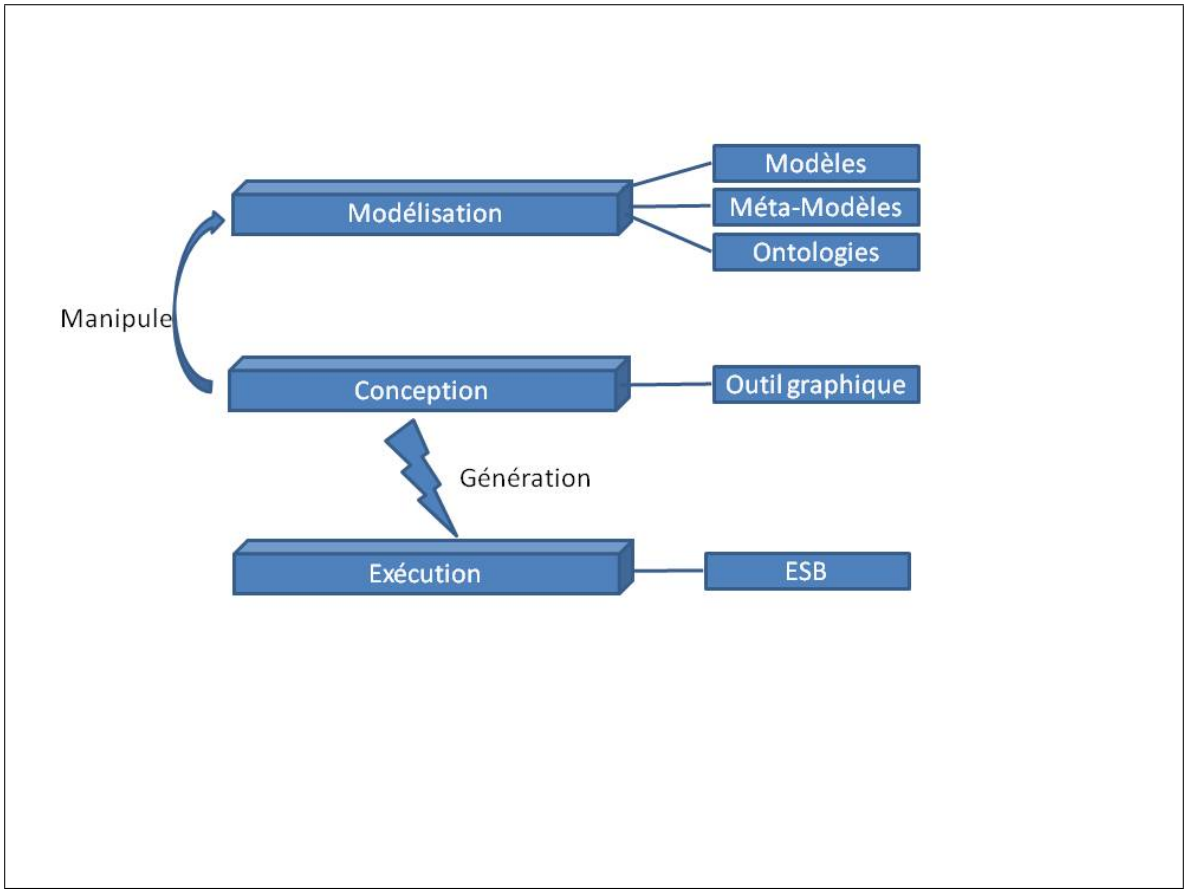

FIGURE 1.2 – Fabrique logicielle

Chapitre 2

Évolution du commerce

2.1 Le commerce du futur

Comme nous l'avons vu en introduction, le futur du commerce est un sujet majeur du pôle de compétitivité des industries du commerce. Le Nord-Pas-de-Calais est une région à forte concentration commerciale. Beaucoup de ces enseignes ont ainsi leur siège social implanté dans la région comme le montre la figure 2.1.

Le commerce a donc dans la région un impact fort sur l'économie par son implantation. Le commerce influe également sur la vie sociale. L'ensemble des enseignes regroupent plus de 20% des emplois de la région. Les enjeux autour de ce domaine sont considérables pour la région.

L'évolution du commerce n'est donc pas un simple problème d'enseigne. Il concerne également toutes les entreprises périphériques, les employés, la région mais également tous les clients, réels et potentiels, sur lesquels ces enseignes ont un impact. En regardant les chiffres de la fréquentation des plus grandes enseignes, on peut ainsi remarquer que les grandes surfaces sont un lieu de vie quasiment quotidien pour de nombreuses personnes. Or, nous avons constaté un écart entre le monde du commerce et les habitudes de vie moderne. Notre vie quotidienne est de plus en plus enrichie par la technologie, notamment les TICs récentes : téléphones portables, Internet...Or encore aujourd'hui le monde du commerce est peu, voire pas du tout, touché par les TICs, que ce soit au sein de la chaîne de valeur que sur les lieux de vente eux-mêmes. Le seul domaine où il y a eu un impact est par définition le commerce électronique, de son début via l'EDI [7] aux dernières années avec les boutiques en lignes [ML99].

2.2 Évolution du commerce

2.2.1 Évolution des consommateurs

Le monde évolue constamment, et la vie quotidienne des individus en est le reflet. Notre époque marque une ouverture sur le monde : grâce aux TICs notamment, nous pouvons être informé des

7. Electronic Data Interchange

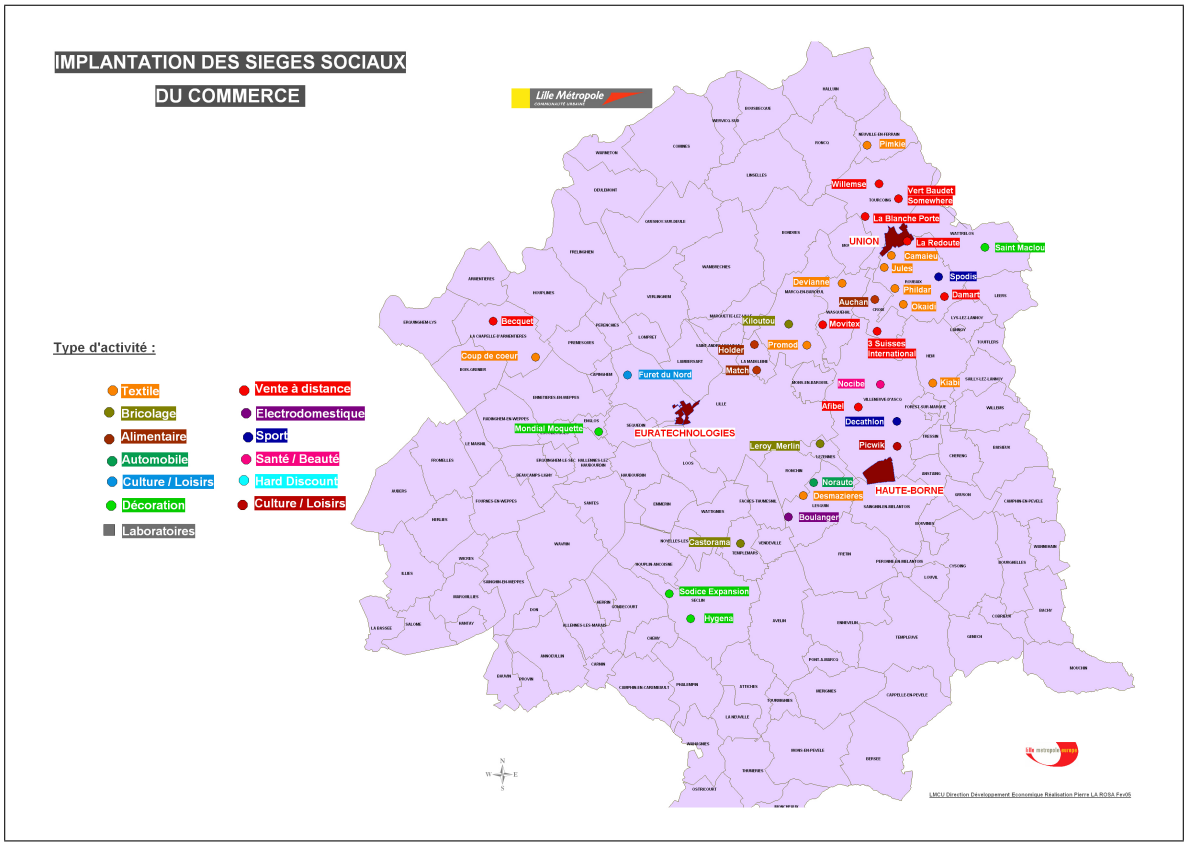

FIGURE 2.1 – Sièges sociaux de la région [REF]

dernières nouveautés, en terme de produits ou par rapport à l'actualité. Les *smartphones* notamment permettent d'accéder à n'importe quel moment à n'importe quelle information via une connexion à Internet. De plus, la multiplication des moyens de communications permet d'être en contact avec n'importe qui en fonction de nos besoins et/ou envie. Les individus évoluant dans ce contexte, il est donc naturel que, en tant que consommateurs, leurs comportements changent également.

A.Derycke, dans un document de prospective sur le commerce a défini les trois grandes caractéristiques des clients à prendre en compte :

- Le comportement utilitariste et plus responsable des consommateurs. Trois grandes catégories de consommateurs se dégagent de cette étude :
  - le client rationnel : le consommateur cherche à moins dépenser en optimisant sa consommation. Pour cela, il est nécessaire de cerner les besoins et attentes du client à l'avance. Ce dernier peut planifier à l'avance son activité de shopping et reproduire une fois sur place le schéma enregistré. Ceci implique une approche qui est dite « Knowledge Intensive » [DH00] y compris côté client. Dans ce cas, l'apport des TICs est nettement visible : le client recherche la connaissance à l'avance, par exemple sur Internet, ou sur place, via un réseau mobile ou via des dispositifs présents sur les lieux de vente comme des bornes interactives.

  - le client gérant son temps partagé : la vie quotidienne évoluant, le temps consacré pour le shopping se réduit. Plus de temps est consacré pour la famille, les loisirs, la communication. . .Il faut donc que le temps des achats soit le plus court possible.
  - le client raisonné : Les consommateurs se préoccupent de plus en plus des aspects éthiques, notamment en ce qui concerne le commerce équitable et le développement durable. Les informations concernant les produits à destination des clients se doivent donc d'être de plus en plus complètes et pertinentes.
- le rôle de l'expérience client : "joindre l'utile à l'agréable". Des études ont montré que les clients de supermarché basent leurs expériences d'achats sur des facteurs dont les principaux sont : ambiance du magasin (température, odeurs, musiques, ...), la qualité de service au sein du magasin, l'image du magasin et les éléments de contextes (l'effet de foule, le temps passé, le prix du panier moyen, ...). Tous ces facteurs conduisent au mécontentement des clients surtout à cause du stress engendré. Sont alors apparus de nouveaux clients, qui sont désintéressés par les courses ou le lèche-vitrine, voire même n'aiment pas cette activité, ils l'endurent plus qu'ils ne l'apprécient. Une des causes de ce *mal-ressenti* est le manque d'innovation dans la façon d'exposer les produits et dans l'interaction avec le client. Les enseignes ne font rien pour réussir à se différencier. Les enseignes doivent donc penser à de nouvelles expérience d'achat pour regagner la confiance des clients. Selon (Lewison, 1997, p. 138), une expérience d'achat peut être ressentie par le maximum d'efficacité (rapidité, prix faibles) ou par le divertissement. Il faut donc que les enseignes arrivent à mieux cerner les besoins des clients et à créer de nouvelles expériences.
- le comportement socialement ambivalent, à la fois plus égocentrique et plus social, des consommateurs

Nous observons depuis des années un comportement général plus égocentrique dans tous les domaines, notamment avec les nouveaux médias. Les individus aiment que l'on parle d'eux, ils veulent de la reconnaissance. Il est donc normal de retrouver un comportement plus égocentrique chez les consommateurs. Ces derniers demandent plus de personnalisation et de différentiation et la reconnaissance en tant que personne. Cette demande a bien été comprise par les enseignes. On observe depuis quelques temps, notamment dans le monde du e-commerce, l'élargissement des approches marketing de type One to One [LV01], que ce soit pour la commercialisation des biens ou des services. Le self-service se développe de plus en plus dans de nombreux espaces de ventes, dont les derniers développements sont la caisse automatique, et *le self-scanning*. L'objectif de cette personnalisation, une forme de prise de pouvoir par le client, est souvent la fidélisation de ce dernier. Le renforcement de la dimension identitaire de la personne est peut-être accéléré par le développement des réseaux sociaux sur le Web. Il faut rappeler que l'essence même de l'Internet, dans ses usages, est d'être un moyen de communication interpersonnel, pair à pair. Le développement du courrier électronique et des messageries instantanées ont amplifié cette propriété fondamentale.

De plus, la socialisation et la recherche de capital social, les communautés de pratiques, les approches ethniques sont une préoccupation de plus en plus importantes pour les individus, notamment pour l'expression de leur identité. La « tribu » devient très importante comme élément de l'estime de soi construite collectivement, comme le succès de FaceBook, Twitter, Viadeo et autres l'atteste.

L'expérience, y compris de shopping, se partage de plus en plus au sein de communauté de pratiques pouvant fortement influencer les comportements de consommation. Cette façon collective de peser sur le monde du commerce a tendance à s'accentuer avec l'évolution des technologies, notamment le web 2.0. Un autre exemple est le réseau social *Ma-résidence* lancé par la mairie d'Evry, en région parisienne [evr]. Le but est de créer un *facebook-like* à l'échelle d'une ville, d'un quartier et d'un voisinage pour développer l'Internet de proximité à l'échelle des écosystèmes locaux.

Enfin, pour conclure, nous pouvons citer le Manifest de Cluetrain [LLSW01] mettant en évidence que les clients ont une meilleure conscience du monde qui les entoure. Ils deviennent donc de plus en plus exigeants.

2.2.2 Évolution des marques et des lieux de vente

Parallèlement aux évolutions des consommateurs, les enseignes ont également évolué, avec des répercussions sur les lieux de vente.

Il apparaît qu'au fil du temps, les TICs ont permis le développement d'une nouvelle forme de commerce, le e-commerce, avec déjà de nombreuses variantes, m-commerce, p-commerce,...(nous y reviendrons dans les sections suivantes) issues des nouvelles technologies, comme le web 2.0, le web sémantique, la communication mobile et les objets nomades. Ces "types de commerces" enrichissent les interactions et les relations avec le client, en "améliorant" son expérience en tant qu'utilisateur et finalement en offrant la perspective de nouvelles stratégies commerciales sophistiquées comme les relations de masse personnalisée (One-to-One).

Ici, il n'est pas question de faire un historique du commerce. De nombreux travaux traitent déjà de ce sujet[ML99]. Nous allons juste montrer l'introduction des TICs dans le commerce depuis quelques années et pointer les derniers projets innovants sur les lieux de vente.

À première vue, il semble que les industries de vente au détail traditionnelles ne sont pas lourdement impactées par les TICs dans leur relation avec les clients. Évidemment les TICs ont une place plus importante qu'il y a une vingtaine d'années, dans des processus de la chaîne d'approvisionnement, ou dans les points de vente pour faciliter le paiement. Mais en réalité une observation attentive de différentes enseignes nous montre que ce n'est pas une généralité et que les TICs sont de plus en plus présentes lors des interactions directes avec les clients, notamment avec la multiplication d'afficheurs électroniques de différents formats : des petits écrans LCD monochromes utilisés comme étiquettes électroniques aux grands écrans utilisés pour la diffusion de publicités dynamiques. Mais nous constatons un manque d'interactivité. Nous pouvons aussi mettre en avant d'autres systèmes, plus interactifs, permettant à un client quelconque de vérifier le prix d'un produit, d'écouter le contenu d'un CD en scannant le code barre, ou encore de sélectionner les bonnes cartouches pour les imprimantes à jet d'encre à l'aide d'une borne tactile interactive située au niveau du rayon concerné. Au vu des récents projets, nous prédisons que les TICS dans les futurs espaces de ventes seront de plus en plus pervasifs. Notre analyse est que premièrement les développements de ces systèmes sont actuellement faits de manières ad-hoc, et deuxièmement que cette évolution doit être coordonnée avec le monde du e-commerce.

En plus des changements conduits par la technologie, nous pouvons aussi mettre en avant une évolution des stratégies commerciales avec notamment une évolution de la relation client. Une tendance des ces dernières années, en relation avec les attentes des consommateurs présentés dans la partie précédente, est la stratégie centrée sur le consommateur : *customer centred* [FM07]. Il s'agit d'étudier les besoins des consommateurs pour mieux privilégier l'expérience du client, permettant ainsi de fidéliser ce dernier en le contentant. Dans certains cas, les entreprises veulent monter un show-room sophistiqué de ces technologies pour deux raisons : savoir comment elles peuvent être utilisées à grande échelle et pour refléter une bonne image, dynamique et innovante, aux yeux du publics. C'est le cas de nombreux projets vitrines d'enseignes célèbres comme Prada dans sa boutique à New-York(Nielsen,Pullin, 2005) ou encore Adidas aux Champs Elysés à Paris, où il est possible d'essayer virtuellement des chaussures de course en surimpression vidéo sur vos pieds en mouvement devant un miroir dédié. Mais c'est aussi le cas de certains "magasins du futur" ouverts par des leaders mondiaux dans la vente comme le "Future Store" de Metro en Allemagne [SSM04]. De nos propres investigations dans ce genre de magasins, il nous a semblé qu'il s'agissait surtout d'un push de technologies et que, à ce stade, il n'a y a pas de vraies interactions riches avec les périphériques mobiles des clients. Mais nous avons identifié un nouvel usage de tags visuels au Japon (QRCode par example), situés sur des surfaces de vente, en relation avec le téléphone mobile du passant pour lui fournir des nouveaux services, nous montrant ainsi un nouveau potentiel d'interaction sur les espaces de vente.

Dans le domaine de la recherche on peut trouver de nombreux projets ayant déjà traité du design et des usages des systèmes informatisés pour le commerce. Très souvent, ils ont exploré le potentiel de technologies mobiles comme le RFID (ou le NFC) attachés par tags individuellement aux produits [KR07] afin de lire rapidement de l'information et d'identifier exactement un produit, et ce avec une interaction plus facile qu'avec un code-barre traditionnel, ou alors ils se basent sur le potentiel des nouveaux appareils mobiles possédés par les clients pour leur fournir plus de services (liste de courses, *self-sanning*, ...). Si les tags RFID gagnent en intérêt auprès des clients et des enseignes, dans la plupart des cas ils sont utilisés sur les lieux de vente pour fournir des nouveaux services aux clients ou améliorer des existants, et pour élargir l'interaction entre le client et le lieu de vente. Dans la littérature scientifique, nous trouvons des propositions d'étagères intelligentes [WW06], de caddies augmentés[Kou03], et des affichages publicitaires intelligents.

L'observation de l'évolution du monde du commerce nous permet d'émettre des hypothèses concernant la consommation et les comportements des consommateurs. La tendance est à l'hybridation, au couplage de différentes formes de commerce, notamment le couplage e-commerce et lieu de vente. Ces mélanges de genre permettent d'apporter aux consommateurs une nouvelle expérience de l'acte d'achat. Certaines enseignes sont partis de cette hypothèse afin de proposer à leurs clients des projets innovants.

2.2.3 Tour d'horizon des marques innovantes

C'est ainsi que les TICs commencent à se diffuser dans les équipements traditionnels de la vente, comme dans les étagères, les vitrines, le caddy, les bornes d'information. La volonté de la grande distribution d'utiliser les technologies RFID, au départ pour la gestion de la chaîne de valeurs, et d'autres

technologies sans contact, notamment pour le paiement, permet de penser que ces nouvelles technologies pourront aussi jouer un rôle important dans le parcours client, dans l'enrichissement de son expérience du lieu de vente. Le potentiel de la Réalité Augmentée commence à être exploité, par exemple, dans des dispositifs comme des miroirs intelligents, pour l'essayage virtuel de lunettes, de rouges à lèvre, de vêtements...Tout ceci conduit à la naissance d'un nouveau palier pour le développement des espaces de vente que nous avons appelé ici e-retail.

Nous allons présenter plusieurs innovations qui touchent différentes "parties" des magasins. Dans un premier temps, nous allons montrer deux exemples d'étagères augmentées. La première, utilisée par Décathlon, permet aux clients d'avoir des informations relatives au produit qu'ils viennent de retirer du meuble. Un écran situé en haut de ce meuble affiche alors une animation relatif au produit. La seconde, souvent utilisée dans les bibliothèques, permet de gérer les stocks de produits présents dans l'étagère. En plus de la détection de la prise ou de la pose de produits, elle permet de contrôler les stocks et d'alerter visuellement en cas de rupture. Elle propose aussi un système de recherche de produits fonctionnant également avec des repères visuels. Ces deux étagères fonctionnent avec une technologie sans contact, le RFID.

La firme Adidas a innové en 2007 avec une nouvelle façon d'essayer ses chaussures de sport au moyen d'un miroir augmenté. Ce projet était présenté dans le magasin situé aux Champs-Elysées à Paris. Il suffit aux clients de se placer devant ce miroir et de choisir le modèle souhaité via un écran tactile. La chaussure choisie apparait en sur-impression sur le pied dans le mirroir. De plus, la projection arrive à suivre les mouvements de pieds des clients. Le miroir est en fait l'image du pied, filmée par caméra, sur laquelle la chaussure est intégrée.

Un autre exemple de miroir magique est aussi présentée au Japon [mir]. Ce miroir sert de stand de maquillage virtuel. Les produits de beauté étant cher, l'interêt d'un tel dispositif est de pouvoir visualisé, en temps réel, tel ou tel maquillage sur notre visage avant de l'acheter.

Nous pouvons également citer les magasins hardRock café qui ont proposé une vitrine augmentée, via la société *Gesturetek*[ges]. Cette vitrine permet de transformer la vitrine traditionnelle en surface tactile pour valoriser les offres disponibles.

Nous allons maintenant montrer deux projets de magasins augmentés. Le Prada Epicenter Store Project[Qui], datant de 2001, proposait à ses clients plusieurs innovations pour impressionner les clients. Les conseillers de vente du magasin était équipés d'outil sans fil leur envoyant des informations pratiques (promotions, nouvel arrivage...). Les ambiances lumineuses et sonores étaient aussi très travaillées. Les produits étaient tous équipés de tags RFID pour les identifier. Les cabines d'essayage avaient aussi été augmentées pour ajouter de l'interaction : reconnaissance du produit, changement de coloris...

Plus récemment, la firme allemande Metro Group propose un magasin servant de vitrine à leurs projets d'innovations. Ce magasin -le Future Store, situé à Toenisvorst en Allemagne, propose aux clients une nouvelle façon de faire ses courses. En plus de proposer un appareil mobile, un Mobile Shopping Assistant, permettant de faire sa liste de courses, tracer son chemin dans le magasin et de faciliter le paiement, il invite le client dans une expérience sensorielle riche : ambiance sonore ciblée et contextuelle au rayon, sol interactif réagissant aux mouvements, diffusion d'odeurs dans certains rayons alimentaires, écrans publicitaires 360°...En plus de cela, un effort est porté sur l'apport d'informations

aux clients via des bornes informative et des robots mobiles. Pour terminer ses courses plus rapidement, le client peut scanner lui-même ses produits pour faciliter le paiement. S'il possède une carte de fidélité, il peut également récupérer des coupons de réductions virtuels qui seront automatiquement crédité lors de la phase de paiement.

Après ce tour d'horizon, nous pouvons établir un classement de ces projets en fonction de leur orientation : supply, chain, orienté client (spécialisation en fonction des attentes des clients : gain de temps, connaissance, expérience unique, réseau social...), orienté vendeur,...

Ce que nous pouvons surtout remarquer dans ces projets, c'est l'absence de prise en compte de l'environnement, c'est à dire qu'aucun dispositif n'a été conçu pour dialoguer avec ce qui l'entoure, que ce soit un autre dispositif, tel un haut-parleur ou une imprimante, ou un objet mobile appartenant à un client, comme un smartphone ou un PDA. De plus, pour la plupart de ces projets, le dispositif a été conçu de manière ad-hoc, pour une enseigne ou marque bien précise et pour une situation particulière (évènement particulier, objets à mettre en valeur non modifiables, magasin précis...). Rien n'a été prévu pour permettre une réutilisation rapide et facile de ces projets, voire même une industrialisation paramétrable.

2.3 Vers le p-commerce

La plupart des projets présentés ci-dessus sont *isolés*. Même regroupés au sein d'un même endroit, comme pour Prada ou Métro, les différents dispositifs ne sont pas sensibles à l'environnement qui les entoure, ils ne communiquent pas entre eux et ils ne peuvent pas être réutilisés facilement dans un autre contexte. Une branche de la recherche se penche sur ces problématiques. Nous appelons cela le p-commerce, pour pervasif. IBM a été un précurseur dans l'introduction de l'informatique pervasive dans le domaine du commerce. Le projet Albatros[Tri99] a ensuite été plus loin dans le développement de ce concept. Le but de ce domaine est d'étendre le champ des environnemnts de communication et d'information dans deux directions : étendre le champ d'utilisation jusque l'habitat des clients via des dispositifs intelligents et de fournir une expérience d'achat plus riche au sein des supermarchés. Cette évolution est rendue possible grâce aux développements de nombreux nouveaux composants technologiques, mais nous en reparlerons dans le prochain chapitre.

2.3.1 Les projets existants

Un des projets les plus significatifs dans ce domaine est MyGrocer de Roussos et al [RTK+02]. Ce projet, datant de 2002, présente un système pour le commerce pervasif. Ce système se veut bi-directionnel : d'une part il se veut à destination du client, en exploitant le potentiel des TICs pour apporter des innovations au client en magasin et même en dehors, et d'autre part il se veut à destination des enseignes de part l'exploitation de données récoltées en magasin, telles les stocks, le nombre d'objets présents en rayon, le taux de retour...

D'un point de vue général, le magasin est équipé de RFID. Si nous prenons le cas d'un client en magasin, le système lui permet d'utiliser par exemple un caddie augmenté, équipé de technologie

RFID et d'un PDA. Ce caddie permet de reconnaître la carte de fidélité du client, lui propose une liste de courses en fonction de ses préférences et de ce qu'il possède encore chez lui, en incluant la date de péremption des produits qu'il a encore. Chaque produit présent dans le magasin possède un tag permettant de l'identifier n'importe où. Les produits posés dans le caddie sont automatiquement reconnus par le système, la liste de courses et la facture se mettent à jour, d'éventuelles promotions associées au produit sont affichées et surtout les données du magasin sont mises à jour (le stock, statistiques relatives au produit...). Le paiement à la caisse est facilité grâce au calcul de la facture en temps réel, un scanning du caddie est néanmoins exécuté par moments. Le PDA incorporé au caddie peut aussi afficher un plan du magasin et définir un trajet pour retrouver un produit.

Mais MyGrocer ne se résume pas à un magasin intelligent. L'environnement global du client est affecté, notamment chez lui. Une fois revenu de ses courses, le client range ses articles dans son mobilier augmenté, lui aussi équipé de lecteurs RFID : étagères, réfrigérateurs, ...Les informations relatives aux nouveaux produits (nature, date limite de consommation) sont alors enregistrés dans le système pour maintenir l'inventaire à jour. De plus, un accès à distance est aussi possible. Par exemple, un client peut avec son téléphone portable consulter son inventaire pour savoir ce qu'il doit racheter. Il peut aussi demander l'aide d'un comparateur de prix qui lui trouvera le supermarché proposant les produits au prix le plus bas. Il peut également passer commande avec son téléphone[KOO08].

Comme l'on peut le voir dans ces scénarios, MyGrocer est surtout orienté à destination du client, il lui facilite la vie. D'un certain côté, il permet aussi aux enseignes une analyse plus fine des transactions en temps réels, afin d'améliorer une partie de la chaîne d'approvisionnement. Par contre à aucun moment, il est question d'expérience sensorielle, de réenchantement de l'univers du supermarché. De plus, à part les clients, les acteurs du commerce ne bénéficient pas d'apports que pourrait amener ce projet. Les vendeurs, magasiniers, chefs de rayon, directeurs de magasin, installateurs à domicile, ..., ne sont pas, ou très peu, concernés par tout ce qu'apporte ce projet. Tout est fait pour tenter d'attirer et de fidéliser le client.

Impulse[YMKM00] est un autre projet à destination des clients. Ce système permet aux client équipés d'un dispositif de type PDA d'ajouter des produits dans une liste, donner des informations relatives au magasin (horaires, prix...). Mais ce projet ne s'arrête pas aux limites d'un magasin. Basé sur un système d'agents communicants, il permet surtout au client de retrouver les meilleurs prix pour les produits recherchés grâce à une communication entre les agents informatisés des différentes boutiques et son propre agent présent sur le PDA. Les agents peuvent même se mettre d'accord et proposer des transactions au client en fonction de sa liste de courses.

2.4 Bilan

L'évolution rapide de certaines technologies permet donc à certaines enseignes d'essayer d'innover. En observant l'histoire, nous avons remarqué que le secteur du commerce de détail est très axé sur la technologie. De nombreuses technologies ont été expérimentées pour rationaliser et optimiser les opérations fondamentales au sein des magasins ou des entrepôts et pour améliorer la chaîne de valeur

dans sa globalité. L'EDI [8], utilisé pour standardiser les processus de commande entre les fournisseurs et les détaillants, et l'introduction de la numérisation de codes-barres en sont les exemples les plus marquants. Pourtant, plusieurs années après l'introduction de systèmes sophistiqués dans la production et la logistique, les chaines de valeurs modernes présentent encore beaucoup de dysfonctionnements entrainant ainsi des baisses de rendements et de productivité, selon des études menés par l'ECR [9] [ecr] en 2001 dans [Eur01]. Ces problèmes se situent aussi bien en amont (problème d'anticipation concernant les niveaux de stocks élevés ou les possibles ruptures de stocks, un taux élevé de retours, longs délais de livraison, ...) qu'en aval (problème de prévisions de la demande, mauvaise planification d'offres promotionnelles, ...). Pourtant avec les technologies actuelles, les derniers développements dans les infrastructures réseau et notamment les réseaux sans fil sans fil, la télévision interactive, les périphériques mobiles et sans fil et les technologies sans fil de récupération automatique de données (RFID [10] par exemple), il est possible d'imaginer et concevoir des systèmes d'information omniprésents pour le commerce. Ces systèmes entièrement intégrés dans l'environnement intégrés seraient capables de ré-enchanter l'expérience du client lors du processus d'achat et ainsi de le fidéliser.

Les nouvelles tendances du commerce mettent en avant des problèmes de mise en oeuvre de ces nouveaux projets. Ces difficultés sont de plusieurs ordres, par exemple, les infrastructures permettant de supporter les projets de type ubiquitaire. La plupart des grandes enseignes fonctionnent avec des systèmes qui commencent à vieillir. Ceci s'explique par la fiabilité de ces systèmes mais aussi par la criticité de ces systèmes. Par exemple, une enseigne ne peut se permettre de changer son système de gestion des caisses sans être sûr de la fiabilité et sans dépenser trop. Une autre difficulté réside dans le déploiement de ce genre de système. Une enseigne doit être capable de déployer un système sur de nombreux sites, parfois dans plusieurs pays différents, et doit donc être capable de personnaliser ce système rapidement et facilement en fonction des cibles. Nos travaux rentrent dans le cadre de l'industrialisation possible de la création de système pervasif dans le contexte du commerce de détail.

8. Electronic Data Interchange
9. Efficient Consumer Response
10. Radio Frequency Identification

Chapitre 3

État de l'art technologique

Nous avons jusqu'à présent abordé les nouveaux projets émergents concernant les lieux de vente et le monde du commerce en général. Nous avons d'ailleurs observé que la plupart de ces nouveaux projets étaient, d'un point du vue développement informatique, élaborés de manière ad hoc. En analysant plus en profondeur ces prototypes réalisés dans le cadre de l'informatique ambiante, nous avons réalisé que derrière ces projets se posent de réels problèmes d'infrastructures inhérents à l'informatique ubiquitaire, avec des besoins croissants en flexibilité, réactivité, contextualisation...Ces besoins sont apparus au fil du temps avec l'évolution des technologies notamment les technologies de l'informatique et des réseaux de communication qui ont permis de résoudre un certain nombre de verrous technologiques. Ce chapitre évoque les évolutions technologiques de ces dernières années qui ont permis d'arriver aux projets présentés dans la partie précédente.

3.1 Intelligence ambiante et informatique pervasive

Avant d'aller plus loin, nous allons définir l'informatique pervasive afin de cerner nos besoins. Nous avons considéré pour nos travaux que les termes informatiques pervasives et informatiques ubiquitaires étaient équivalents. De nombreux travaux de recherche traitent déjà de ce sujet, nous allons ici juste résumer les grandes caractéristiques. Le concept d'**informatique pervasive** a été établit pas Weiser en 1991 [Wei99] lorsqu'il a pour la première fois introduit le terme "ubiquitaire" pour l'informatique. Les concepts clés de l'informatique pervasive sont :

- en tous lieux, notamment grâce au triomphe du sans fil (connectivité et mobilité généralisées)
- en tous moments grâce à la miniaturisation des objets communicants permettant d'y avoir accès n'importe quand dans n'importe quelle situation (accessibilité permanente)
- le tout sans rupture, avec continuité (seamlessness) quelques soient les circonstances, mais en prenant éventuellement en compte les différents contextes, utilisateurs, physiques, et technologiques, pour y adapter dynamiquement les contenus informationnels ainsi que les interfaces des usagers avec les services. C'est ce qui est appelé l'informatique dite context-aware

– avec l'apparition de nouvelles formes d'interactions homme-machine : interfaces tangibles, réalité augmentée, multimodalité (vocale, gestuelle, ...), ...

Il faut savoir que derrière ces principes fondateurs des différentes formes nouvelles d'informatique, il y a deux autres propriétés qui sont induites :

– la géolocalisation de l'utilisateur, ou plutôt de son dispositif mobile , car afin d'offrir une continuité de services lors de la mobilité , il faut savoir où il se trouve. À partir de cette fonctionnalité interne aux réseaux de communications on peut offrir des services géolocalisés
– *l'atteignabilité* permanente est déduite de l'accessibilité permanente

L'intelligence ambiante vise les mêmes buts mais en mobilisant les apports scientifiques et technologiques de l'Intelligence Artificielle (IA) pour en faire des systèmes intelligents pouvant anticiper sur nos besoins, voire nos désirs en tant qu'utilisateur.

De notre point de vue, ces concepts proviennent des avancées technologiques de plusieurs domaines. Nous ne négligeons pas non plus le côté interaction homme-machine : il est bien évident que les méthodes pour s'interfacer avec de tels systèmes ne se limitent pas aux outils "traditionnels" tels le clavier, la souris et l'écran.

3.2 Évolutions des appareils mobiles

Depuis plusieurs années, nous avons pu remarquer une nette évolution dans les appareils mobiles. Récemment, les téléphones se sont d'ailleurs transformés en "smartphones" : les objets mobiles communicants, les « mobiles » dans le langage courant, sont multifonctions, de véritables « couteaux suisses » de la communication et de l'accès à l'information.

Nous pouvons noter les évolutions suivantes :

– l'évolution du GSM vers la 4G, comme avec le standard technologique LTE [11] proposé par la GSM Association [LTE]. Les technologies sans fil offrent un gros débit pour les informations numériques, et proposant de plus une convergence entre réseaux longue distance, et réseaux de proximité type Wifi ou Bluetooth, et autres proches du GSM à très courte distance, notamment au sein de la maison ou dans le magasin, avec les technologies de type NFC ...
– l'évolution des écrans de visualisation : le rapport résolution/taille s'améliore notamment avec les technologies OLED
– l'objet nomade communicant devient un capteur de son environnement et un lecteur universel de médias : de son (recherche d'une musique que l'on est en train d'entendre dans le voisinage), d'images fixes et animées, de positions (H/V), de mouvements pouvant être des gestes, de proximité d'objets numériques (via les technologies RFID ou NFC) de lecteurs visuels d'étiquettes (exemple code barre plus prix)
– de plus en plus de mobiles compatibles NFC sont destinés à être utiliser comme moyens de paiement.

11. Long Term Evolution

Il faut pourtant nuancer les avancées dans ce domaine. Au vu du marché actuel, il semble qu'il y aura sans doute encore pendant plusieurs années une forte hétérogénéité dans les plates-formes logicielles sous-jacentes aux mobiles (systèmes de Nokia, Microsoft, Apple, Android de Google, ...) et des guerres de standards technologiques ou de modes de diffusion des nouvelles applications, via le téléchargement plus ou moins contrôlé comme dans l'Appstore. Cela conduira les concepteurs de nouveaux services à des démarches de conception de type « développement durable », qui devraient survivre aux incessantes évolutions technologiques en offrant une forme de plasticité [CC02], d'adaptabilité dynamique, d'interface Homme-Machine, et de production des services et des contenus informationnels visant un très large public.

3.3 L'informatique pervasive

La multiplication de petits dispositifs mobiles de plus en plus puissants et de moins en moins chers, reliés ensemble par un seul et même réseau, et l'apparition de logiciels qui s'adaptent à leur environnement ont conduit à l'informatique dite "pervasive". D'après [SBP+08], cette informatique se traduit par le fait que partout, l'environnement qui nous entoure est composé d'équipements intelligents, communicants et parfaitement adaptés aux utilisateurs. Pour créer une application pervasive, il faut donc réussir à combiner tout ou partie de ses dispositifs tout en prenant en compte les propriétés suivantes :

- l'hétérogénéité de l'environnement
- la dynamicité de l'environnement
- les contraintes des dispositifs
- les besoins de l'utilisateur

Concernant la dynamicité de l'environnement, nous avons regardé les travaux concernant les réseaux de capteurs, couplés à des outils sémantiques et permettant de suivre en temps réel l'évolution de l'environnement.

3.3.1 Le contexte

Dans les environnements pervasifs, le contexte est un mot clé essentiel. Mais ce n'est pas toujours facile d'en avoir une définition exacte. Nous en présentons quelques unes que nous retrouvons dans la littérature afin de mieux cerner cette notion. Nous montrerons ensuite différents moyens de traiter avec le contexte.

3.3.1.1 Définitions

Nous avons retenu cinq principales définitions du contexte dans le monde pervasif.

La première définition que nous retrouvons dans la littérature date de 1994 et est émise par Schilit [SAW94] : location of use, the collection of nearby people and objects, as well as the changes to those objects over time". Cette définition nous présente le contexte comme étant le lieu où l'utilisateur se trouve, ainsi que les objets et personnes proches. La notion de proximité est mise en avant pour souligner les éléments caractéristiques du contexte. Il est également souligné que les changements sur

les éléments présents au fil du temps font partie du contexte. Cela sous-entend que le contexte n'est pas fixe et peut donc varier en fonction du temps.

La seconde définition date de 1997 et est émise par Brown et al ; [BBC97]. Selon eux, le contexte est caractérisé par l'utilisateur et les éléments de l'environnement physiques de l'utilisateur. Ces éléments peuvent être l'heure, la saison, la température, l'identité de l'utilisateur, sa localisation ...

Toujours en 1997, nous retrouvons les mêmes fondements émis par Ryan et Pascoe [RPM97]. Le contexte se résume à quatre éléments principaux : l'environnement, la localisation, le temps (au sens chronologie) et l'identité de l'utilisateur.

Nous pouvons remarquer que dans les deux cas, le contexte comprend l'environnement de l'utilisateur et l'utilisateur lui-même. Il est partie intégrante de son contexte. Il est à noter également que le contexte n'est pas clairement défini comme un ensemble fini d'éléments. Brown et al. donnent des pistes et des exemples mais pas une liste exhaustives et fixes d'éléments pouvant être inclus dans le contexte. L'environnement, pour reprendre le terme de Ryan et Pascoe n'est pas clairement défini. Nous pouvons également remarquer que la temporalité est toujours présente dans les deux définitions : l'évolution temporelle fait partie du contexte.

En 1999, des travaux [ADB+99] poussent la réflexion plus loin et formalisent les notions que nous avons présentées. Leur définition est la suivante :

context is any information that can be used to characterize the situation of an entity. An entity is a person, or object that us considered relevant to the interaction between a user and an application, including the user and the application themselves Ces travaux donnent en effet un nom à la notion de changement au fil du temps : le contexte est lié à la notion de situation. Une situation est l'état d'un contexte à un instant donné. Ils caractérisent aussi les éléments de l'environnement pouvant être inclus dans le contexte comme étant tous des personnes ou des objets utiles. Cette caractéristique sous entend une forme de subjectivité et "non-réutilisabilité" du contexte : ce dernier est construit pour une application et correspond à un point de vue, celui des créateurs de l'application.

Enfin, [TC02] parlent eux de la notion de "Contexte d'interaction" qui se caractérise par un triplet : <plate-forme - environnement - utilisateur >. Pour eux, l'environnement est l'ensemble des entités périphériques à la tâche courante et pouvant impacter le système ou l'utilisateur. Cette fois ci, les éléments doivent impacter le système ou l'utilisateur. Il est alors possible de dresser une liste non subjective de ces éléments pour une application donnée.

En 2001, Dourish [Dou01] apporte un élément supplémentaire. Selon lui, l'activité de l'utilisateur est un élément à prendre en compte dans le contexte.

Dans nos travaux nous avons alors retenu les caractéristiques suivantes pour définir le contexte :

- le contexte est constitué de tous les éléments, objets, personnes, applications impactant une application, y compris l'utilisateur et l'application elle-même
- le contexte est dynamique : l'état et même la présence des éléments évoluent au fil du temps
- les éléments "possibles" du contexte sont déterminés à l'avance et connus par le système

Comme nous venons de le voir, la notion de contexte n'est pas facile à déterminer et donc encore moins à manipuler. Néanmoins, il existe des classifications du contexte. Ces classifications permettent

un découpage logique du contexte pour mieux cerner les éléments concernés. Nous avons retenu la principale émise par Shilit en 1994.

Dans ces travaux, le contexte est découpé en trois sous-ensembles :

- le contexte utilisateur : profil, localisation, situation sociale
- le contexte informatique : connexion réseau, coûts de communication
- le contexte physique : luminosité, température, pression, heure ...

Chaque élément du contexte peut se classer dans l'un de ces sous-ensembles : l'heure se retrouve dans le contexte physique, le périphérique utilisé dans le contexte informatique...Ces sous-ensembles conditionnement le traitement des éléments qu'ils contiennent. De plus, ils donnent une idée de la façon dont ces éléments vont impacter l'application.

Dans nos travaux, nous nous sommes axés principalement sur le contexte informatique et sur le contexte utilisateur, notamment son environnement de travail.

3.3.1.2 Capture des contextes

Un des éléments essentiels de la prise en compte du contexte réside en la capture de celui ci : comment récupérer les informations qui nous intéressent concernant le contexte de l'utilisateur. Et surtout, comment se tenir informé des changements survenus sur ces éléments, le plus rapidement possible en cas d'effet de bord sur l'application et son utilisation. La capture du contexte passe essentiellement par une collecte de données envoyées par un ensemble de dispositifs (capteurs, appareils mobiles, ...). Cet ensemble de dispositifs est appelé *source de contexte*[DA00]. Au sein d'un environnement pervasif, les sources artificielles de contexte peuvent être des capteurs (pression, thermomètre, capteur de présence, ...), des périphériques, des logiciels, ...tout élément pouvant donner des informations.

Les applications dites "context-aware" doivent donc réussir à gérer la complexité induite par la capture du contexte : réussir à récupérer, trier et interpréter un ensemble de données d'origines hétérogènes. Pour réaliser cette opération, la notion de "Gestionnaire de contexte" a été introduite. Ce gestionnaire traite la capture du contexte en trois phases :

1. la déclaration ou la découverte des sources de contexte
2. l'abonnement aux sources
3. l'envoi de notifications diffusant les informations des sources

La phase 1 permet à l'application de connaître à l'exécution les éléments sources de contexte. Il existe deux façons de procéder :

1. la découverte : le gestionnaire de contexte balaie l'environnement pour découvrir des périphériques pertinents
2. la déclaration : l'élément source vient se déclarer auprès du gestionnaire comme étant une source de contexte

Une fois les éléments connus, le gestionnaire met en place deux mécanismes permettant de rester au courant de la situation des éléments. Le premier est donc l'abonnement aux sources. Le gestionnaire peut s'inscrire auprès des sources de contexte pour être tenu au courant de l'évolution.

Enfin, un mécanisme de notification permet de recevoir en temps réel le ou les changements subis par une source de contexte.

L'implémentation de ce genre de gestionnaire pose de sérieux problèmes en terme d'architecture. Il faut pouvoir s'abstraire de l'hétérogénéité, pouvoir récupérer les "bons" dispositifs en fonction du contexte et enfin pouvoir reconfigurer dynamiquement une application selon le contexte et en prenant en compte les besoins de l'utilisateur. Pour ce genre de problème, il est nécessaire d'avoir une architecture souple et flexible et dans laquelle il est possible d'avoir des informations sur les différents modules présents. Nous nous sommes penchés sur les travaux récents sur les architectures et notamment sur les architectures orientées services. Ces recherches se sont basées sur les limites des approches orientées objets. Ces limites concernent d'une part la contextualisation : un objet ne possède pas de description contextuelle de son état à l'exécution. Cette limite restreint l'usage des objets pour une utilisation dans un environnement pervasif, où l'état des objets doit être connu de tous en en fonction du contexte doit pouvoir agir sur l'environnement. D'autre part, l'approche objet limite également la flexibilité d'une application, de par la dépendance forte entre objets.

3.3.2 Conception d'un environnement pervasive : l'approche orientée services

L'informatique pervasive nécessite des besoins forts en adaptabilité, modularité et flexibilité. Ces notions caractérisent la possibilité de modifier le comportement d'un système en fonction de son contexte.

C'est pourquoi nous nous intéressons à l'approche orientée services (SOA pour Service-Oriented Architectures). Dans cette approche, une application est vue comme un ensemble de services dialoguant entre eux par des messages. Le service forme alors le bloc de base d'une architecture SOA. Cette approche définit le service comme élément de base d'une application. Nous donnons plus loin la définition d'un service. Dans cette approche, nous distinguons trois acteurs : un fournisseur de services, un consommateur de services et un médiateur, ou facilitateur, entre les deux, appelé registre ou encore annuaire de services. Le schéma de la **figure 3.1** présente le lien entre ces trois acteurs. Le fournisseur peut publier son service, notamment dans un annuaire contenant le contrat, et le consommateur peut alors via cet annuaire découvrir le service. L'annuaire de services référence l'ensemble des services (et des contrats associés) disponibles au sein du contexte d'utilisation de l'application. Une fois le service retrouvé, le consommateur peut récupérer une référence vers ce service pour ensuite l'invoquer. La dynamicité d'une telle architecture se traduit par la possibilité laissée aux fournisseurs et consommateurs de quitter ou de rejoindre le système au cours de l'exécution.

Le W3C nous propose la définition suivante concernant les architectures orientées services : ne architecture orientée services est un ensemble de composants pouvant être invoqués, et dont les descriptions d'interfaces peuvent être publiées et découvertes.

La notion de services se retrouve dans plusieurs définitions :

– Bieber[BC01] : service is a contractually defined behaviour that can be implemented and provided by any component for use by any component, based solely on the contract.

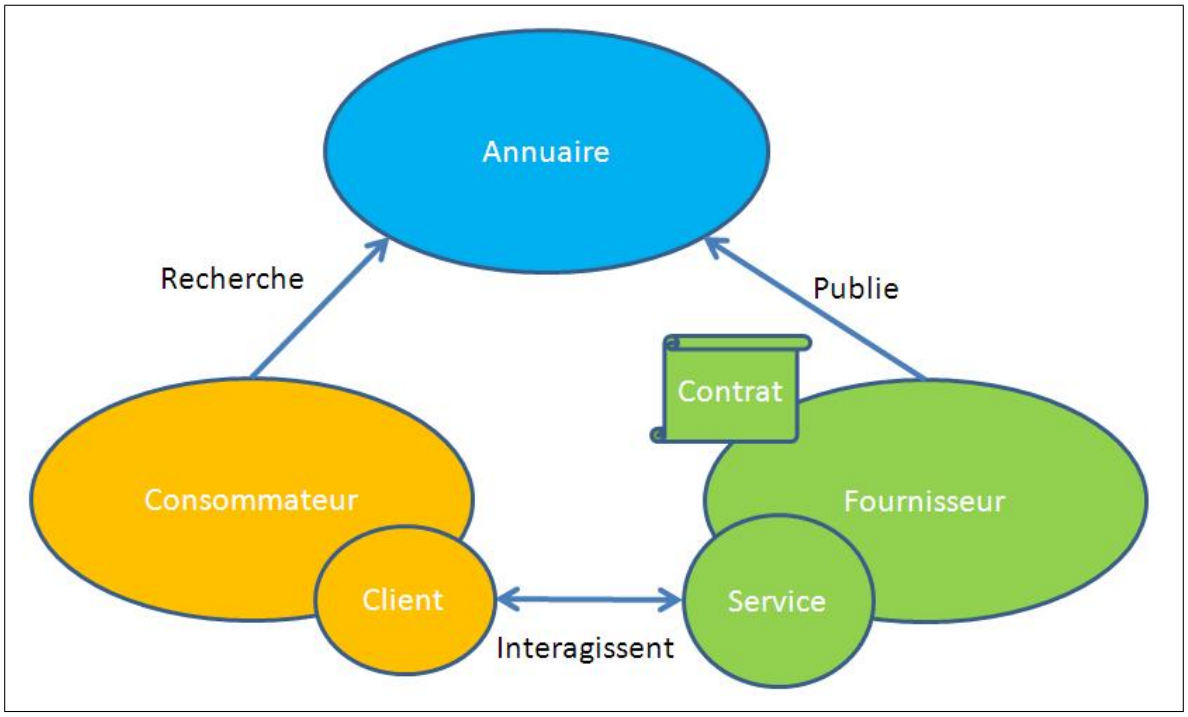

FIGURE 3.1 – Architecture SOA

– OASIS[sera] : service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description. A service is provided by an entity the service provider for use by others, but the eventual consumers of the service may not be known to the service provider and may demonstrate users of the service beyond the scope originally conceived by the provider.
– Papazoglou[PTDL07] : ervices are autonomous, platform-independant entities that can be described, published, discovered, and loosely coupled in novel ways. They perform functions that range from answering simple requests to executing sophisticated business processes requiring peer-to-peer relationships among multiple layers of service consumers and providers.

Si on regroupe toutes ces informations, nous pouvons sortir les principales caractéristiques d'un service :

– le service doit :
  – respecter un ensemble de contrats (interfaces, règles de fonctionnement)
  – offrir un ensemble d'opérations dont les interfaces sont publiées
  – être autonome (disposer de toutes les informations nécessaires à son exécution : pas de notion d'état)
– le service peut :
  – être codé dans n'importe quel langage

– s'exécuter sur n'importe quelle plate-forme (matérielle et logicielle)

Un service est donc autonome dans le sens où il ne dépend d'aucun contexte ou service externe. Il est constitué d'opérations représentant les actions spécifiques que le service propose.

Le contrat d'un service permet à un utilisateur et au fournisseur de services de s'entendre sur l'utilisation de ce service. Un contrat contient en effet des règles pouvant être négociées entre le fournisseur et l'utilisateur. Ces règles sont appelées SLA [12]. Un contrat peut contenir plusieurs SLA. Ces règles peuvent par exemple porter sur les temps moyen de réponse, le temps de disponibilité, ...

Une architecture orientée services est donc composé d'un ensemble de services qui communiquent entre eux et peuvent interagir afin de créer un service de plus haut niveau. Cette communication se base sur des échanges de messages, synchrones ou asynchrones. Les différents services peuvent interagir entre eux mais le couplage entre services est un couplage faible afin de garder la propriété d'autonomie d'un service. Cet assemblage de services est appelé *composition de services*. Une composition de services est créée à partir d'une orchestration de services. Une orchestration de service décrit la collaboration entre des services pour aboutir au but recherché. Une orchestration décrit les interactions d'un service avev d'autres services.

Ces interactions peuvent être décrites via plusieurs langages dont les principaux sont :

– BPEL [13] [bpe] : langage XML permettant de définir des processus basés sur des appels de services
– BPML [14][bpm] : spécification pour un langage de description de processus orientés services, basé sur XPDL

Nous pouvons alors enrichir la description d'un service avec les caractéristiques nécessaires à la composition :

– une granularité étendue (coarse-grained) : contrairement à un composant technique (électronique par exemple), un service propose des opérations permettant de remplir plusieurs fonctions. Un service doit aussi pouvoir traiter un large périmètre de données et ne pas être trop spécifique, afin de conserver une certaine réutilisabilité
– un interfaçage souple : un service peut implémenter plusieurs interfaces. Et à l'inverse plusieurs services peuvent implémenter une même interface
– une identité : Par identité, nous entendons une localisation, une description et une unicité d'instance. Avant d'appeler un service, il faut le localiser. De plus, un service possède une instance unique. A la différence des composants qui sont instanciés à la demande et peuvent avoir plusieurs instances en même temps, un service est unique.
– un Couplage lâche ou faible (loosely-coupled) : les services sont connectés aux clients et autres services via des standards. Ces standards assurent le découplage, c'est-à-dire la réduction des dépendances. Ces standards sont des documents XML comme dans les web services
– une communication synchrone ou asynchrone

Un exemple d'implémentation de services est par exemple le service Web qui utilise WSDL (un méta langage XML) comme langage de description, un annuaire UDDI pour en permettre la localisation

12. Service Level Agreement
13. Business Process Execution Language
14. Business Process Modeling/Management Language

et un protocole de transport comme http dans l'architecture REST et SOAP pour l'architecture SOA. De plus, les webs services possèdent leur propre langage de composition basé sur BPEL *BPEL4WS*[15] .

La problématique d'assemblage de services dans le but de créer une application, le tout dans un environnement dynamique, est à l'origine de l'approche dite *à composants orientés services*. Cette approche vise à définir l'architecture d'une application bâtie sur un assemblage de services. Le service est englobé dans un élément appelé *composant*. Le composant embarque alors des informations facilitant l'assemblage et la réutilisation des services. Les principales informations sont des descripteurs de dépendances un nommage fort des composants. L'implémentation de ce modèle le plus répandu est la SCA[16] [Spé] proposée par IBM. Cette architecture renforce le découplage entre le service et son implémentation.

3.3.3 Approches dérivées de SOA

L'architecture orientée services a fortement contribué à l'apparition de nouvelles architectures d'applications. Pour nos travaux, nous nous sommes penchées sur deux concepts en particuliers :

- le cloud computing
- le mash up

Le Cloud Computing[clo] fait référence à l'utilisation de la mémoire et des capacités de calcul des ordinateurs et des serveurs répartis dans le monde entier et liés par un réseau (principe de la grille informatique). Une application se retrouve donc répartie dans le monde entier et peut être utilisée par qui le demande.

Le SYNTEC Informatique, dans son livre blanc du Cloud Computing[clo], présente trois modèles différents de ce type d'architecture : IaaS, Paas, SaaS.

- Infrastructure As A Service (IaaS) est le premier modèle de cloud, où
  - l'entreprise maintient : les applications, les runtimes, l'intégration SOA, les bases de données, le logiciel serveur
  - le fournisseur Cloud maintient : la virtualisation, le matériel serveur, le stockage, les réseaux
- Platform As A Service (PaaS) est le second modèle de cloud, où :
  - l'entreprise maintient uniquement les applications.
  - le fournisseur Cloud maintient : les runtimes, l'intégration SOA, les bases de données, le logiciel serveur, la virtualisation, le matériel serveur, le stockage, les réseaux
- Software As A Service (SaaS) est l'ultime modèle de cloud, où le fournisseur Cloud maintient : les applications, les runtimes, l'intégration SOA, les bases de données, le logiciel serveur, la virtualisation, le matériel serveur, le stockage, les réseaux. Le SaaS peut être vu comme un modèle économique de consommation des applications : celles-ci sont consommées et payées à la demande (par utilisateur et par minute d'utilisation par exemple) et non plus acquises par

15. Business Process Execution Language for Web Services
16. Service Component Architecture

l'achat de licences. Le SaaS peut donc à ce titre reposer sur une infrastructure informatique dans le nuage.

Une autre approche dérivée de la SOA est le **mash up**. Le **mash up**, ou application composite, est une application qui combine du contenu ou du service provenant de plusieurs applications plus ou moins hétérogènes [FJRT07]. Dans le cas de site Web, le principe d'un mashup est donc d'agréger du contenu provenant d'autres sites, afin de créer un site nouveau. De plus en plus d'éditeurs de contenu proposent gratuitement des API, afin d'encourager la communauté des développeurs à créer des mashup utilisant leur contenu. C'est le cas de Google, de Yahoo !, de Amazon,...via des APIs web services, leur intérêt étant d'inciter les développeurs à répandre et à diffuser leur contenu. Les applications composites se basent sur la programmation événementielle[GTPL09].

Dans notre cas, nous voulons créer des environnements intelligents en proposant de relier des dispositifs entre eux. Nous proposons alors de définir ces dispositifs comme des fournisseurs de services.

Un des problèmes qui se posent concernent la provenance des dispositifs. Les environnements intelligents peuvent contenir des composants provenant de fournisseurs divers. Les architectures citées précédemment ne règlent pas le problème de l'infrastructure technique à mettre en place pour répondre aux différentes contraintes suivantes : besoin de supporter différents protocoles ou piles de communications, besoin de masquer le réseau...

Dans la littérature, les composants appelés Intergiciels permettent de régler tout ou partie de ces problèmes.

3.3.3.1 Les intergiciels

Dans un système pervasif, des tâches concurrentes peuvent être exécutées sur des dispositifs distribuées et dans un environnement hétérogène. Le logiciel médiateur, ou intergiciel, permet donc de masquer cette hétérogénéité et même de masquer le caractère distribué d'une application de part sa position comme indiqué dans la **figure 3.2**. Une application peut donc en appeler une autre (via un appel de méthodes, un envoi de message, une invocation de services...) sans avoir besoin de connaître ni la localisation ni même la nature (application de bureau, base de données, ...) de celle ci.

Pour supporter l'hétérogénéité, nous avons donc besoin d'un intergiciel qui offrent un niveau d'abstraction tel qu'il est possible de relier les dispositifs entre eux. Un logiciel médiateur, terme introduit dans le monde informatique depuis environ 1968 [NRB68], est un logiciel reliant une ou plusieurs applications. Il permet notamment le transfert de données d'une application à l'autre. C'est pour cela qu'on peut aussi appeler ce logiciel un logiciel d'intégration de données. Un logiciel médiateur se fonde sur trois techniques :

- l'appel de procédure à distance
- l'échange de messages
- la manipulation d'objets tiers

Pour cela, un tel logiciel fournit un ensemble de fonctionnalités de haut niveau :

- localisation transparente d'une application ou d'un service au travers d'un réseau
- filtrage, transformation des données

- indépendance vis à vis du réseau
- fiabilité et disponibilité
- extensibilité

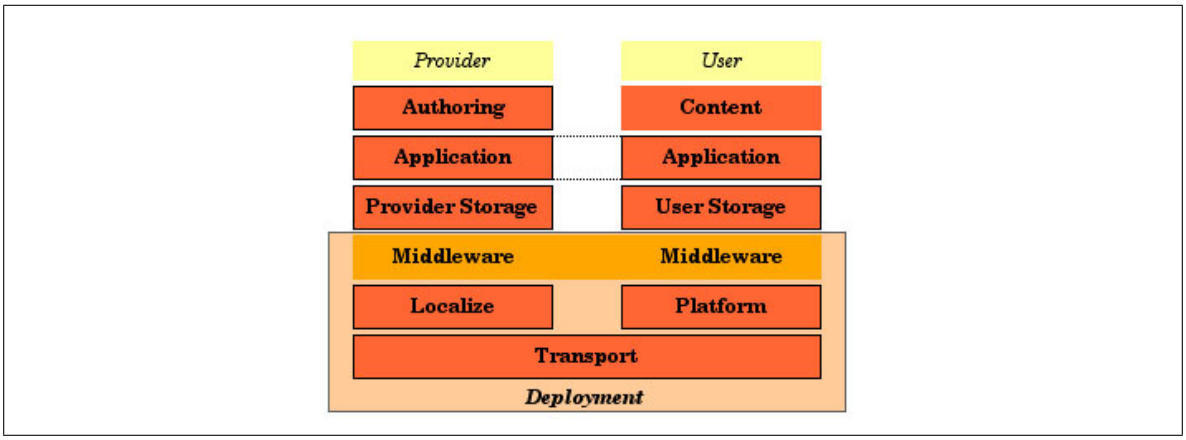

FIGURE 3.2 – Positionnement de la couche intergiciel dans le modèle OSI

Il existe plusieurs sortes d'intergiciels. Judith Hurwitz propose un classement dans son article [Hur98]. Le système de classification de Hurwitz présente les différents types d'intergiciels qui sont disponibles. Ces classifications sont basées sur l'extensibilité et la fiabilité :

- appels de procédures à distance : le client fait des appels à des procédures situées sur des systèmes distants. Ces appels peuvent être synchrones ou asynchrones.
- les intergiciels orientés message : les différents acteurs (client et services) interagissent via des envois et réceptions de messages
- object Request Broker : Ce type intergiciel permet aux applications d'envoyer des requêtes et des objets au sein d'un système orienté objet.
- SQL-oriented Data Access : Intergiciels dédiés aux communications enter applications et serveurs de base de données.
- embedded Middleware : services de communication et interface d'intégration software/firmware qui opère entre des applications embarquées et des systèmes d'exploitation en temps réel.

On peut étendre cette classification en ajoutant d'autres caractéristiques :

- réflexifs : on entend par réflexivité la capacité de l'intergiciel de raisonner et de tirer ses propres interprétations concernant notamment la façon de dialoguer entre différentes applications.
- orientés composants : les intergiciels à base de composants se basent sur les paradigmes de la SCA
- orientés services : les intergiciels orientés services répondent aux caractéristiques décrites précédemment concernant la SOA.

Il existe actuellement plusieurs plates-formes de médiation dont les principales sont présentées ci-dessous.

Corba CORBA permet l'interaction de services distribués via un bus. Un registre référence les services mis à disposition. Le consommateur doit obtenir une référence vers le service correspondant à son besoin. CORBA peut être implémenté sur différents langages. Mais ce système est assez ancien et ne dispose pas de services de notifications, ce qui est un frein à la dynamicité d'un système.

UPnP UPnP permet la création de réseau spontanés. via la transparence de l'infrastructure et de la configuration réseau. Les services sont fournis par des dispositifs connectés sur le réseau, auto-configurables et auto-desciptifs. La pile UPnP est indépendante du langage de programmation. Un mécanisme de notification, réalisé directement par les dispositifs, permet de connaître les arrivées, les départs et les changements d'états des différents services . De plus chaque composant doit maintenir son propre registre de services qu'il a lui même découvert sur le réseau. UPnP est décrit plus tard dans le chapitre 4.

Fractal Fractal est un modèle de composants ouvert et adaptable permettant de construire des applications via des assemblages de composants. Fractal est une implémentation de la norme SCA. Dans ce modèle, un composant est inclus dans un élément appelé *membrane* qui possède les propriétés de contrôle. Les composants peuvent être assemblés via des fichiers de description *ADL*[17]. Un composant possède plusieurs interfaces :

- **interface serveur** : définit les services fournis par le composant
- **interface client** : définit les services requis par le composant
- **interface de contrôle** : définit les besoins et les informations non fonctionnels du composant (sécurité, tolérance aux pannes, règles de gestion, ...)

Fractal possède des implémentations en différents langages :

- en C
- en Java
- en C++
- en SmallTalk
- en .Net

ReMMoC : Refflective Middleware for Mobile Computing ReMMoC est un intergiciel réflexif[GBS03]. Il permet de résoudre le problème d'hétérogénéité entre intergiciels. Il peut être considéré comme un intergiciel d'intergiciels. Il permet le développement d'applications indépendamment des technologies. Cet intergiciel fournit deux services de haut niveau :

- un service de découverte de services
- un service de correspondance de données

17. Assembly Description Langage

Ces services permettent de configurer dynamiquement les applications en permettant le changement de services via le service de correspondance de données.

AURA : Towards distraction free pervasive computing. AURA [AUR] définit une architecture permettant de réaliser des tâches quotidiennes représentées comme des applications abstraites, de manière transparente, sans configurer manuellement les différentes applications. Les tâches utilisateurs sont composées de services abstraits présents dans l'environnement. Un des points forts de AURA est l'adaptation automatiques des tâches en fonction des ressources disponibles dans l'environnement. De plus, chaque environnement est capable de reconfigurer l'exécution des tâches en respectant les capacités des services et les ressources disponibles. AURA s'intéresse donc principalement à l'adaptation dynamique de l'environnement en manipulant les tâches utilisateurs.

OSGi OSGi est une implémentation SOA locale : il s'agit d'une plate-forme orientée services, dynamique et se basant sur un annuaire de services local. OSGi est orienté JAVA et possède un mécanisme de notifications pour tenir les services à jour concernant l'évolution du contexte d'exécution. OSGi est décrit également dans le chapitre 4.

Les différentes plates-formes présentées possèdent donc leurs propres caractéristiques et leurs propres limites. Par exemple, certaines sont mono-langage, d'autres ne proposent pas de découverte automatique de services. Nous pouvons alors suivre deux approches. La première serait d'implémenter notre plate-forme sans prendre en compte les logiciels médiateurs présentés ci-avant. La seconde serait de coupler certaines solutions, notamment OSGi et UPnP. Nous avons pour nos travaux décidé de suivre cette deuxième approche, comme nous le présenterons dans le **chapitre 5**.

L'informatique pervasive est le sujet de plusieurs travaux de recherche et certains projets résultent de ces travaux. Nous allons maintenant présenter certains de ces projets.

3.4 Projets relatifs à l'informatique pervasive

3.4.1 Amigo

Le projet européen Amigo - Ambient Intelligence for the networked home environment, est un projet de recherche en domotique. L'objectif principal de ce projet de recherche est la mise au point et le développement d'un nouveau type de middleware open source, standardisé, et permettant la plus large inter-opérabilité entre des services répartis dans une maison et connectés entre eux à l'aide d'un réseau. À partir de ce middleware, de nouvelles pistes doivent être explorées afin de résoudre les problèmes d'Interopérabilité qui surviennent régulièrement entre différents constructeurs.

L'intergiciel Amigo offre des fonctionnalités permettant de découvrir des services présents sur le réseau (Service Discovery) et de les faire interagir. Les échanges de données entre ces services web sont assurés par le protocole SOAP. En complément de ce noyau, de nombreux services ont été développés pour définir un cadre logiciel autour de la maison intelligente. L'interopérabilité entre les différents

fournisseurs de services et d'équipements est assuré grâce à la définition d'une ontologie commune et l'utilisation de la même sémantique à tout niveau. Amigo prend également en compte le contexte de l'utilisateur, via son service *Context Management Service*. Ce service est chargé de collecter et de redistribuer des informations liées au contexte parmi les composants du système Amigo. Toute donnée recueillie, ou produite par un service ou composant, peut-être assimilée à une information de contexte. Tout composant Amigo peut alors devenir une source de contexte. Ces services, sources de contexte, peuvent être interrogés de manière synchrone (query-driven), en appel explicite, ou asynchrone (data-driven) par système de notification. Dans le cas d'une demande synchrone, le langage de requête ontologique SPARQL du W3C est utilisé[spa]. Dans le cas asynchrone, le consommateur d'informations contextuelles doit s'abonner auprès de la source et lorsqu'une donnée est générée, elle est automatiquement envoyée auprès du consommateur, et auprès de tous les consommateurs ayant souscris à cette source. Amigo est aussi un projet portant sur les usages notamment sur la construction de scénarios exploitant les réseaux IP domestiques. Les membres du groupe ont organisé un challenge tournant autour du projet. Ce challenge consistait à proposer des applications et à les développer à l'aide des briques logicielles Amigo.

3.4.2 Activity-Based Computing project

Le projet ABC[Bar05], débuté en 2001, concerne l'intégration de l'informatique pervasive dans le contexte de support au travail dans les hôpitaux. Il résulte d'un processus de création centré utilisateur. Il permet aux utilisateurs de gérer leurs dispositifs, services et ressources comme des activités informatiques en permettant de les créer facilement, les partager et en permettant de changer le contexte d'exécution à la demande, et ce quel que soit le dispositif utilisé.

3.4.3 Wcomp

WComp[WCO11] est un environnement de prototypages d'applications orientées services. La particularité de cette plate-forme est la sensibilité au contexte, notamment l'adaptation en fonction des dispositifs disponibles dans l'environnement. Cette plate-forme propose l'exécution de ces application via les containers, composants englobants les dispositifs et leurs services, mais également la création via des éditeurs appelés designers. Basés sur des modèles, ces éditeurs permettent une approche multi créateur afin d'adapter les applications à plusieurs représentations (en fonction des rôles des utilisateurs par exemple). Les technologies utilisées par cette plate-forme sont UPnP et DPWS. La communication est événementielle. La création d'une application s'effectue par une orchestration des services proposés par les dispositifs de l'environnement. WComp vise principalement le développement facile et rapide de prototypes d'applications multi-devices pour la domotique.

Basé sur cette plate-forme, l'Ubiquarium[ubi] est un environnement expérimental de tests permettant de simuler des interactions avec des objets réels (capteurs, boutons relais, ...) et des objets virtuels intégrés dans un environnements 3D comme le montre la **figure 3.3**. L'Ubiquarium est constitué de divers dispositifs ainsi que de services *découvrables* et composables dynamiquement. Ces dispositifs peuvent être soit des dispositifs virtuels (objets d'une scène 3D dans laquelle l'utilisateur est im-

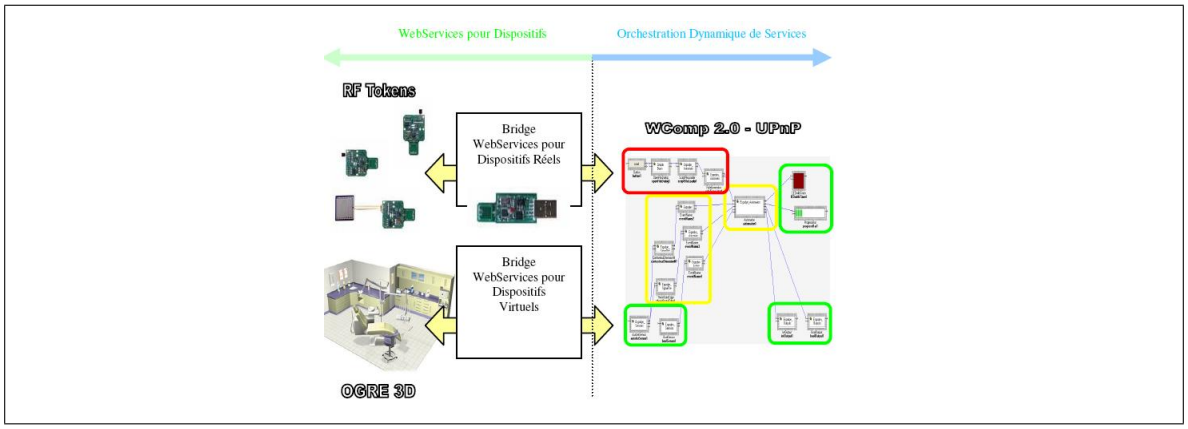

FIGURE 3.3 – Vue logique de la plate-forme WComp

mergé), soit des dispositifs réels portés par l'utilisateur ou présents dans son environnement. Tous les équipements de l'Ubiquarium, réels ou virtuels, sont donc basés sur une interface de type Web Service (SOAP, UPnP, DPWS...) ce qui limite ainsi au maximum les conceptions ad-hoc. L'Ubiquarium actuellement mis en œuvre repose sur trois grandes classes d'équipements :

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

Cet environnement permet alors l'évaluation de nouvelles applications de l'informatique mobile et ambiante telles que les usages des ordinateurs portés ou « Wearable Computers ». Il permet aussi l'étude des mécanismes d'adaptation logicielle pour des applications sensibles au contexte.

Par exemple, des éléments d'une scène 3D, une télévision dans la **figure 3.4**, un PC portable et d'autres appareils multimédia, représentés comme des services UPnP, peuvent être contrôlés via des contrôleurs physiques réels tels que des boutons.

3.4.4 Bilan

Cette revue de projets nous permet de mettre en évidence les faiblesses de ces projets. Par exemple, si nous prenons en compte l'hétérogénéité, tous les projets ne proposent pas la possibilité d'interagir avec n'importe quel dispositif, langage, technologie ...

Une autre limitation de ces projets concerne l'outillage fourni pour la création d'un environnement intelligent. Ni Amigo ni Wcomp ne fournissent d'interface utilisateur de haut niveau permettant à n'importe qui de générer un tel environnement.

FIGURE 3.4 – Exemple de scène 3D utilisée dans WComp

Nous avons alors étudier les travaux traitant d'interfaces utilisateurs simples dédiées à la conception de systèmes informatisés.

3.5 Nouvelles façons de concevoir des applications

Nous allons conclure cette partie par un bref aperçu des nouveaux moyens mis à disposition des utilisateurs afin de concevoir des systèmes informatisés de façon simple et compréhensible. Un des buts recherchés est la possibilité de créer un programme informatique sans écrire une ligne de code. Le logiciel Scratch [SCR] en est une parfaite illustration. L'objectif affiché de ce genre de logiciel est de rendre accessible la programmation à des débutants, y compris des enfants. Ce genre d'outil permet de masquer le plus possible la partie technique d'un programme informatique. L'utilisateur ne choisit pas un langage, des classes ou des méthodes, il n'écrit pas non plus de lignes de commandes pour créer du comportement. Nous allons ici présenter deux de ces logiciels : Scratch et Alice [ALI].

3.5.1 Scratch

Scratch est un outil qui permet à son utilisateur de créer des programmes informatique en manipulant non pas un langage informatiques mais des objets graphiques qu'il aura créés au préalable. À ces objets sont associés des actions qui peuvent être comparées à des méthodes comme définies dans un langage objet. L'utilisateur n'écrit pas de lignes de code mais manipule ces objets, les associe, tisse des liens entre eux afin de créer des comportements. Il ne sait pas qu'il écrit de la logique. Le logiciel s'occupe de générer le programme qui correspond aux objets et comportements "dessinés" par le créateur. Pour faciliter la compréhension, l'IHM est conçue afin de faciliter l'utilisation : des codes de couleurs sont associées aux actions et/ou objets, les icônes sont auto explicites...[SCR]. La **figure 3.5** présente une copie d'écran de ce logiciel.

Scratch fournit aux utilisateurs des blocs visuels de programmation pour construire la logique et le flux de données d'un programme. Il suffit alors pour créer un programme d'emboiter des blocs. Ces blocs possèdent différentes formes et couleurs afin d'éviter les erreurs de syntaxe et de type de données. Les programmes sont alors crées à partir de piles de blocs. Ce logiciel permet à l'utilisateur de manipuler des structures de programmation comme les boucles, les conditions et la programmation évènementielle sans le savoir.

Scratch est écrit en Squeak (langage) et est open-source.

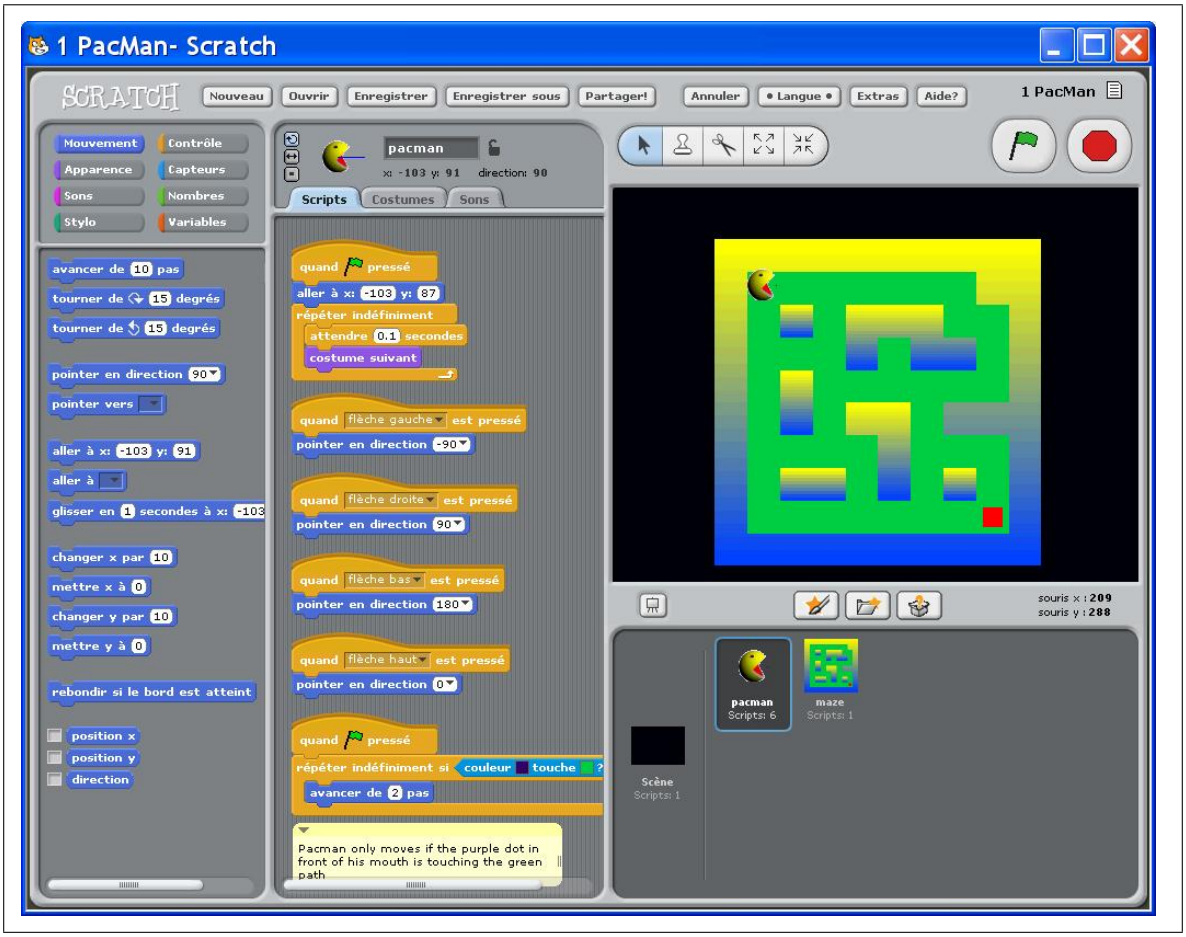

FIGURE 3.5 – Environnement de développement Scratch

Scratch est issu du projet Lego Mindstorms.[min]

3.5.2 Alice

Un autre projet de ce type est Alice, du nom du personnage créé par Lewis Carrol. Il se présente sous la forme d'un environnement de programmation en trois dimensions dans lequel l'utilisateur peut déplacer et déposer (drag-and-drop) des objets 3D et insérer comme dans scratch des briques

permettant d'ajouter du comportement. Ces briques représentent des déclarations informatiques. Alice permet de créer des jeux interactifs, des animations et des vidéos. L'environnement interactif permet de manipuler et tester directement les objets et les programmes créés. Alice a été pensée pour être une introduction à la programmation orientée objets et pour permettre à des débutants, informaticiens ou non, d'apprendre les bases de la logique, de l'automatisation et du multimédia. Comme le montre la copie d'écran de la **figure 3.6**, le logiciel fourni est moins intuitif que Scratch et se base sur un système phrasé, contrairement à Scratch qui était plus graphique avec son système de briques. Alice a été développée par une équipe de recherche dirigée par Randy Pausch. Alice a été conçue selon trois principes : théorie d'apprentissage de l'informatique, le support du développement orienté objet et de la programmation évènementielle, et la volonté d'encourager le storytelling. Alice est également un projet opensource.

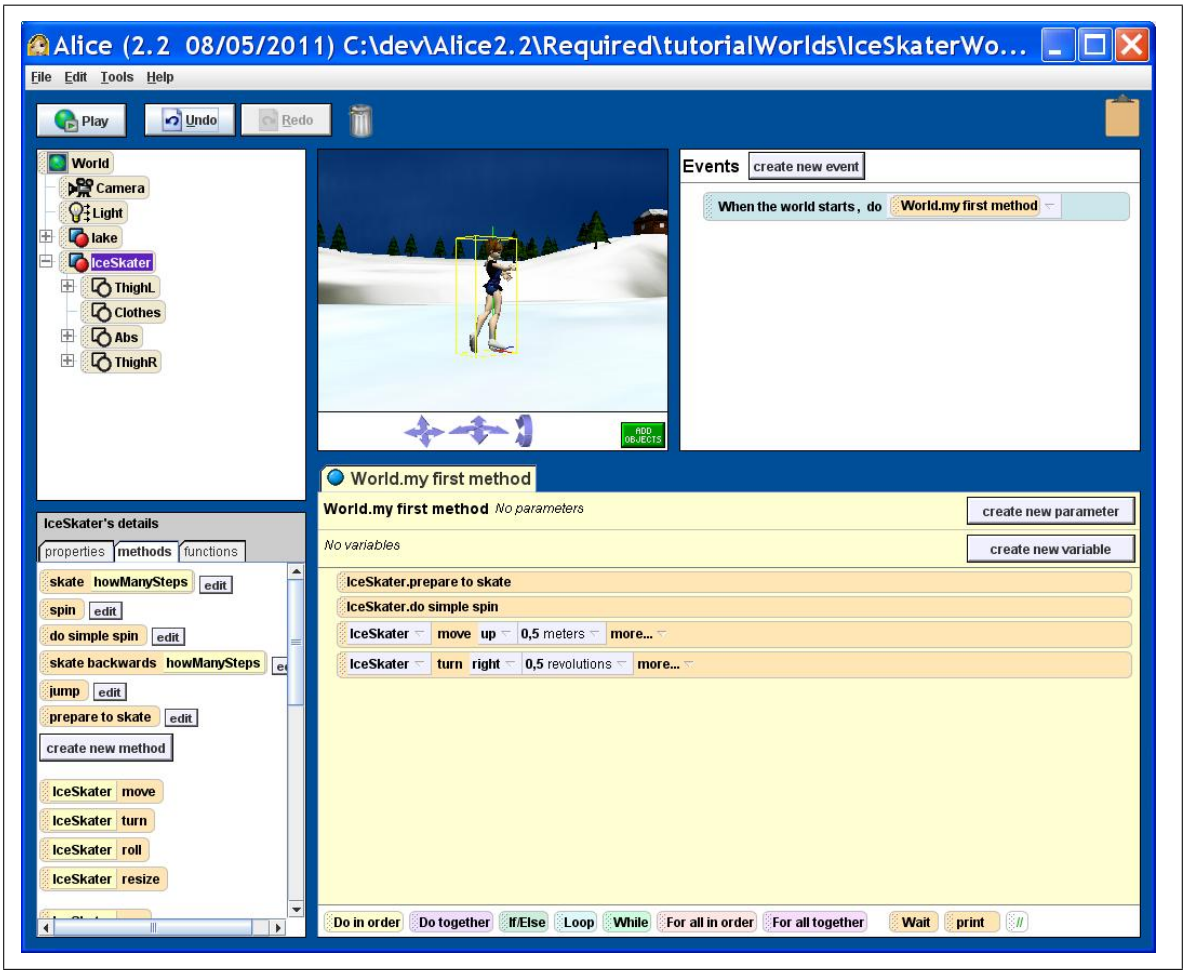

FIGURE 3.6 – Environnement de développement Alice

3.6 Conclusion

Ce chapitre nous permet donc de mettre en lumière plusieurs points. Le premier concerne le contexte. Nous avons des besoins en termes d'infrastructure pour capturer, traduire et utiliser aux mieux les informations fournies par le contexte. Comme nous l'avons indiqué, des projets traitent déjà de ce sujet, mais de manière spécifique, liée au domaine d'application dédiée (la maison pour le projet Amigo par exemple). Une telle infrastructure doit reposer sur un socle technique permettant la communication entre plusieurs applications ou périphériques. Les intergiciels décrits dans ce chapitre vont nous servir de base de réflexion pour la constitution d'un socle technique. Un second point important concerne la possibilité d'améliorer les interactions hommes machine, en terme d'expérience utilisateur, via l'apport de la réalité augmentée. Le dernier point abordé concerne les travaux autour de la conception de systèmes informatiques par des personnes non spécialisées. Ce point nous semble important à aborder car un des principaux challenges de nos travaux est la création rapide d'environnements pervasifs. En se basant sur les projets cités dans ce chapitre, nous avons imaginé un logiciel pour simplifier la génération d'environnements de vente intelligents.

Chapitre 4

Problématiques de notre recherche et ses domaines d'application

4.1 Intelligence ambiante

L'intelligence ambiante est au coeur de nos préoccupations. Permettre l'accès au ressources disponibles dans l'environnement et ce n'importe où, n'importe quand...nécessite de nombreux besoins, en terme d'infrastructure et de logiciels informatiques, et pose alors beaucoup de contraintes. Nous avons notamment pu observer via l'état de l'art que les projets liés au commerce sont réalisés de manière "ad hoc", sans ambition industrielle, la complexité de tels systèmes nécessitant trop de moyens de mise en œuvre à grande échelle. En plus des problématiques liés à l'intelligence ambiante, la démocratisation de ces systèmes informatiques passe par une meilleure capacité de production de ce genre de système. Un coût trop élevé de mise en place d'un système est un frein à son installation et donc son utilisation. Un problème d'industrialisation se pose donc. De plus les problématiques liées à l'*utilisabilité* se posent également. Pour rappel, l'*utilisabilité*, défini par la norme ISO 9241-11 comme :

> le degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié

Pour résumer, le système doit être facile à utiliser, efficace et attirant malgré la complexité de mise en oeuvre.

Nous nous intéresserons aussi à nos terrains d'expérimentations avec les besoins spécifiques liés au p-learning et au commerce.

4.2 Industrialisation

Dans la partie précédente, nous avons dressé le constat que les projets utilisant l'intelligence ambiante sont développés de manière ad hoc, ce domaine souffrant d'un problème d'industrialisation.

Une question peut alors légitimement se poser : pourquoi une entreprise choisirait d'installer un ou plusieurs systèmes avec de l'intelligence ambiante si cela lui coûte trop cher. Il est donc nécessaire de proposer un système qui permet un retour sur investissement intéressant. Cette contrainte entraine des conditions sur la mise en œuvre et les services rendus par le système souhaité. Nous avons abordé dans la partie précédente la caractéristique *utilisabilité* du système. Cette caractéristique a pour objectif de susciter un intérêt des clients. Si nous prenons la perspective d'un lieu de vente, dans un supermarché par exemple, l'*utilisabilité* d'un tel système a pour but de réattirer les clients au sein des magasins, à l'époque du e-commerce, et de les fidéliser. Ce point fait partie intégrante de la relation client.

Nous allons aborder maintenant la partie industrialisation. Pour rendre ces systèmes rentables, la plus value apportée par l'intérêt des clients ne suffit pas (dans la plupart des cas), la production et la mise en place doivent être les moins coûteuses possibles.

4.2.1 Production

Les stratégies économiques des entreprises sont en train de changer. Sans entrer dans les détails, elles sont passées de la production individuelle à la production de masse. Mais depuis une dizaine d'années, les tendances s'inversent, les clients veulent pouvoir personnaliser leurs produits. Le concept de *production de masse personnalisée* est très récent et illustre bien ce nouveau choix stratégique.

Ce concept consiste à rendre personnel, adapter et individualiser l'offre de masse. Cela peut être considéré comme une technique de production permettant de fabriquer en grandes quantités des produits qui conservent chacun des caractéristiques spécifiques (couleurs, inscriptions,...), correspondant aux attentes des clients, et ce afin de répondre plus précisément à la demande des clients.un des problèmes majeurs est de réussir à maîtriser les coûts de la personnalisation sur un nombre important de systèmes. Mais il reste un problème lié à la conception car si le coût de production a souvent chuté, le coût de conception lui reste constant, voire augmente dans certains cas.

4.2.2 Mise en place et intégration de solutions

La mise en place d'un système informatique ubiquitaire, notamment son intégration au sein d'un SI existant, nécessitent aussi un travail conséquent. Dans le contexte de développement informatique, l'intégration d'un tel système nécessite une parfaite cohésion entre les différents protagonistes du projet afin de respecter les différentes contraintes : délais, budget et niveau de qualité. Mais les outils font encore défaut. Il n'existe aucune suite de développement logiciel intégrant des outils de collaboration et de gestion de projet. Dans les projets actuels, ces problèmes se traduisent par un taux d'échec record dans le domaine des projets informatiques dans l'industrie : seuls 16 % des projets tiennent leurs engagements, tandis que 37 % sont purement et simplement arrêtés en cours de route. Les 47 % restants dépassent les délais ou le budget initialement prévus. (Source :Borland).

Les entreprises sont cependant nombreuses à n'avoir ni les compétences techniques ni le temps pour assembler des plates-formes de développement collaborative et multi-acteurs. De plus, pour éviter des surcouts à chaque nouveau projet, il faut penser à la réutilisation de cette plate-forme pour plusieurs clients, dans notre cas, plusieurs magasins. Une solution consiste à découper leurs ateliers de dévelop-

pement en un ensemble de modules. Ceux-ci sont fédérés, côté serveur, par un socle d'intégration et, côté client, par un environnement de développement aisé personnalisable.

Si nous nous intéressons au socle d'intégration, depuis plusieurs années, plusieurs travaux ont été menés afin de faciliter l'intégration de modules au sein des plates-formes d'entreprise. Les Enterprise Integration Pattern sont un résultat significatif de ces travaux. Les EIP, comme le montre la **4.1**, définissent un ensemble de patrons de conception, les *design pattern* permettant l'intégration d'application au sein des entreprises. Ces patterns sont issus des besoins de couplages faibles entre applications au sein d'une entreprise.

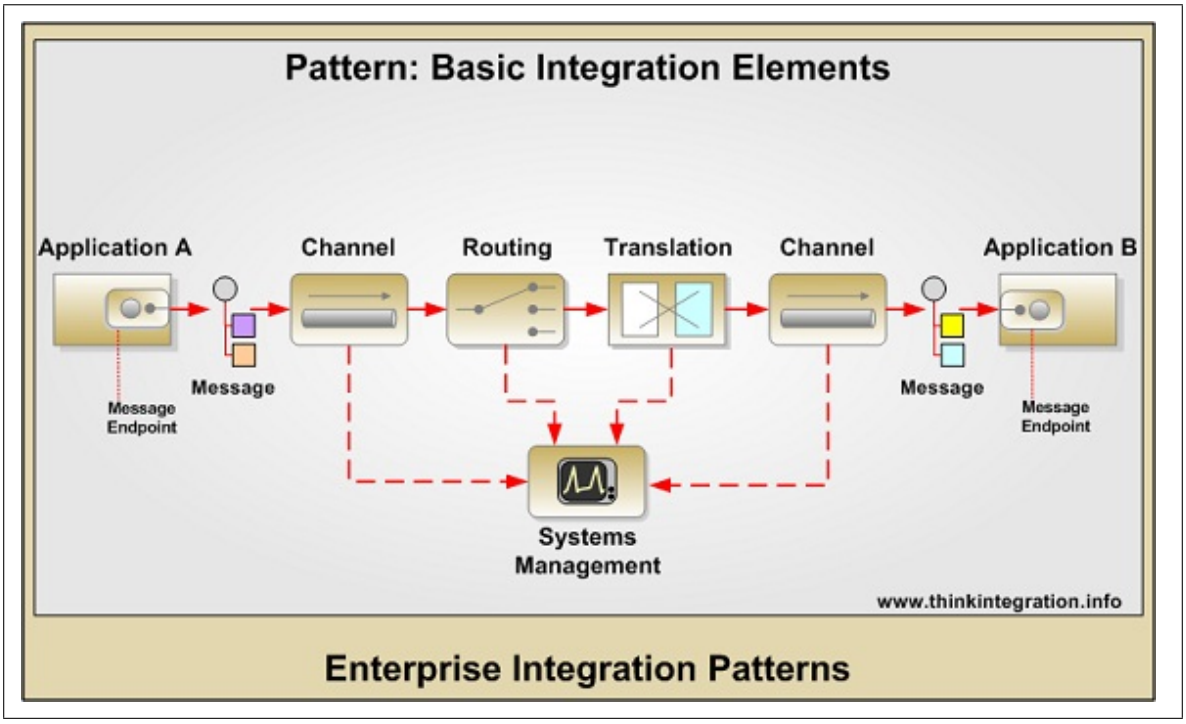

FIGURE 4.1 – Principaux éléments de l'intégration orientée message

Les trois principaux éléments de l'intégration orientée message sont :

- Message Bus : architecture type permettant à des applications séparées de travailler ensemble mais de façon découplée afin que les applications puissent facilement être enlevées ou ajoutées sans affecter les autres
- Event Message : permet de transmettre des évènements d'une application à une autre
- Message Router : permet de transmettre des messages en fonction d'un ensemble de conditions

Nous avons retenu trois types d'intégration définis dans les EIP.

Fonctionnalités métier partagées De nombreuses applications métiers implémentent des fonctionnalités qui existent déjà ailleurs. De nombreux systèmes ré-implémentent leurs propres fonctions pour vérifier la validité d'une adresse, ou dans le domaine du retail, pour connaître le stock d'un

produit. Ce patron d'intégration propose alors d'implémenter ces fonctionnalités une seule fois et les rendre disponibles comme un service à tous les systèmes en ayant besoin.

Architectures orientées services Nous avons déjà traité de SOA dans le chapitre 3. La SOA nous permet de créer des applications complètes à partir de fonctions métier partagées exposées comme services.

Processus métier distribués Un processus métier distribué est un processus business qui permet de coordonner l'exécution de fonctionnalités métiers s'exécutant sur plusieurs systèmes.

4.3 Besoins non fonctionnels

D'autres besoins apparaissent également avec l'apparition de ce type de projet.

4.3.1 Étude des usages

Le premier est apparu avec l'essor d'Internet. Lee, considéré comme le créateur d'Internet, disait qu'il est impossible de prévoir ce que les gens vont faire d'Internet. Les gens créent leurs propres usages. Il en va de même avec l'apparition des nouvelles technologies. Il est nécessaire d'étudier les usages que les gens en font afin :

- de connaitre les préférences des utilisateurs pour améliorer le système ou la personnalisation du système (notion de profil utilisateur)
- d'identifier de nouveaux usages induits par l'utilisation faite par les utilisateurs

Nous disposons de plus en plus de moyens pour étudier les usages. Certains sont intrusifs (eye tracking, caméra, ...) mais permettent une étude en utilisation réelle des systèmes. D'autres moyens se réalisent à la fin de l'activité étudiée, comme les questionnaires ou des séances de debriefing.

4.3.2 Personnalisation

La notion de personnalisation est à rapprocher du concept économique dit du *B2C*[18]. Ce concept définit les relations entre une entreprise et ses clients. La personnalisation est une demande de plus en plus forte de la part des entreprises et des clients. Les entreprises veulent utiliser un produit qui soit à leur image, et les clients souhaitent utiliser des produits qui correspondent à leurs envies et leurs gouts. Ils veulent vivre une expérience agréable. La personnalisation est donc double pour le créateur du système : il doit permettre à une entreprise ou une marque de personnaliser les EVi en fonction de l'enseigne et de l'environnement associé à cette enseigne (logo, charte graphique, ...) mais également permettre une configuration à la volée pour qu'un client puisse aussi le personnaliser à l'utilisation.

18. business to client

4.3.3 Méthodologie de projet

Nous avons pu remarquer au cours de nos expérimentations que l'informatique ubiquitaire, comme la plupart des projets innovants, se heurte à un manque d'ouverture des entreprises, notamment les équipes dédiées (DSIO). Les entreprises possèdent leurs propres protocoles d'intégration de nouveaux projets. Ces protocoles sont mis en place depuis des années. Mais les nouveaux projets ne rentrent plus dans le cadre de ces anciens protocoles. Il arrive souvent qu'il faille créer des prototypes dont la durée de vie est éphémère, et autour desquels il faut mettre en place une infrastructure particulière : étude des comportement, étude des usages, enquête de satisfaction, réunion de co-conception ...De plus, l'informatique ubiquitaire utilise des technologies, protocoles, matériels, ...récents et dont les utilisations ne sont pas approuvées par les services informatiques des entreprises.

Un dernier problème concerne la concurrence. Les entreprises ne sont pas encore toutes prêtes à divulguer les résultats d'expérimentation de tels projets à d'autres entreprises. Ce comportement empêche une capitalisation qui permettrait de gagner du temps sur les conceptions des systèmes et surtout sur l'étude des utilisateurs.

4.4 Pervasive learning

Le projet p-LearNet, présenté en introduction est à l'origine des travaux présentés dans cette thèse. Ce projet tente d'utiliser les apports de l'informatique pervasive afin de proposer de nouvelles façons d'apporter de la connaissance pour les professionnels et ce pendant leur travail. Dans ce projet, nous avons collaboré avec des entreprises nous servant de terrain d'expérimentation. L'un de ces terrains est une enseigne de supermarché à rayonnement national avec laquelle nous avons réfléchi à l'apprentissage de ses vendeurs pendant leurs heures de travail et au sein de leurs rayons. La problématique de cette entreprise était le bilan de l'utilisation de leur plate-forme d'apprentissage actuelle : celle-ci était peu utilisée car non pratique. Les vendeurs pour accéder doivent se rendre sur des machines dédiées en dehors de leurs rayons. Pour eux, ce temps d'apprentissage correspond à un risque de rater des ventes et donc de voir leurs revenus baisser, étant payés sur commissions. Nous avons donc réfléchi à une solution d'apprentissage in situ : en temps réel, n'importe où et surtout n'importe quand, pouvant être interrompue à tout moment pour être reprise plus tard. Nous avons pour cela d'abord étudié les techniques d'apprentissage actuelles. Les techniques de formation professionnelle s'inspirent peu à peu des nouvelles techniques issues des sciences de l'éducation : de nombreuses plates-formes de formation en e-learning ont vu le jour par exemple. Mais il reste un écart entre ces deux mondes. Le monde du travail semble s'être satisfait du e-learning sans essayer les nouvelles techniques d'apprentissage naissantes : Micro-Learning, blended Learning, ...

De nombreux projets autour de l'apprentissage mobile, dont MyartSpace [VMS+06, VSR+09], mettent en avant l'utilisation de périphériques mobiles et du travail collaboratif pour améliorer l'apprentissage grâce au pervasif. Dans le contexte d'une situation de travail, nous retrouvons des contraintes au niveau des besoins : nécessité de transparence, de rapidité. L'apprentissage ne doit pas être une contrainte et pour cela il ne doit pas être dissocié du travail de l'apprenant. Dans de telles conditions,

il faut pouvoir apporter à l'apprenant en situation de travail les ressources, services,...nécessaires à cet apprentissage, tout en essayant de l'assister dans son travail. Notre volonté est donc d'apporter une méthode d'apprentissage qui devienne également un support à l'activité du travailleur. De plus il n'existe pas d'interaction entre ces différentes méthodes. Pour allier apprentissage et aide dans un contexte pervasif, nous proposons un assistant personnel informatisé à destination des vendeurs, le PSA pour Personal Selling Assistant. Pour préciser les concepts, nous parlons bien ici d'assistance, il n'est pas question de remplacer le travailleur ou de l'obliger à modifier sa fonction.

4.4.1 Assistant personnel à la formation

Cet assistant, sous la forme d'un périphérique mobile, permet aux vendeurs d'accéder à un ensemble de services concernant leur formation ou de l'aide à la gestion de leurs produits : récupération des stocks, des prix, de la marge réalisée par les vendeurs sur un produit...Nous développons plusieurs fonctions pour le PSA pour permettre aux vendeurs de se former ou de réviser ses connaissances en juste à temps, en utilisant les ressources de l'environnement. Le PSA peut être étendu avec des périphériques présents dans le rayon du vendeur (haut-parleurs, écrans LCD, ...) ou des services pour par exemple lire des code barre, ou des tags RFID. Cela permet au vendeur d'accroître la collaboration avec le client en partageant les ressources dont il dispose. Pour améliorer l'efficacité des vendeurs et les aider dans leur travail, le PSA peut fournir plusieurs services : l'accès et l'échange de connaissances concernant les produits ou services vendus, la possibilité de partager les connaissances dans une communauté de pratique,...L'arrivée du WEB2.0 a permis l'apparition d'outils comme les Wikis ou les Blogs, couplés à un système de base de cas répondant à ces besoins. Ces mêmes outils, utilisant donc l'apport sémantique du Web 2.0, peuvent fournir au vendeur des conseils pour sa vente pendant celle-ci, en présence du client. L'utilisation de la multimodalité peut s'avérer judicieuse dans ce cas. Par exemple l'utilisation d'une sortie vocale de synthèse, via des écouteurs, peut aussi fournir du coaching personnalisé au vendeur, en temps réel et devant le client. De même, la mise en oeuvre d'une interface multimodale utilisant de la voix en entrée, avec le standard VoiceXML par exemple,développée dans un framework pour le e-commerce, Ubi-Learn [CDR06] ou la conception d'un coach digital permet également la création d'un coaching personnalisé à destination du vendeur.

4.4.2 Assistant personnel à la vente

Après une première expérimentation de notre PSA, nous avons décidé d'étendre les possibilités de ce dernier en lui ajoutant des fonctionnalités dédiées au support pendant l'acte de vente. En plus des capacités cités précédemment, celui-ci devient une vraie télécommande permettant au vendeur de contrôler l'environnement intelligent dans lequel il évolue. Il devient plus qu'un outil, il permet d'étendre le champ d'application du vendeur en générant un BAN [19]. Le vendeur devient communicant au sein de son espace de vente intelligent. Il peut avoir accès et paramétrer tous les périphériques qui sont à sa disposition : écran, haut-parleurs, imprimantes, ...et ce via des technologies dédiées (UPnP, DLNA, ...). De plus, le processus de vente est clairement intégré dans l'outil. Le client est intégré dans

19. body area network

l'espace de vente intelligent. Le vendeur peut à tout moment, et peu importe où il se trouve dans le magasin, être averti de la présence d'un ou plusieurs clients et de connaître à l'avance ce à quoi ces personnes s'intéressent, via des capteurs présents dans le rayon. La fonction communicante de l'outil permet également au vendeur d'échanger des informations avec l'appareil mobile d'un client pendant une discussion.

4.5 Résumé des problématiques

Avant d'aller plus loin, il est important de résumer les thématiques abordées dans nos travaux. Comme décrit précédemment, nos problématiques sont de plusieurs ordres. De par la nature même de nos recherches, nous devons répondre aux contraintes posées par l'intelligence ambiante : prise en compte du contexte, accès aux ressources n'importe quand, n'importe où et depuis n'importe quel périphérique. Nous avons axé nos recherche sur la question suivante : comment faire interagir des objets, périphériques, systèmes d'information, humains, ...de manière transparente via un système informatisé ? Derrière cette question, se posent d'autres problèmes : quels types d'interactions, avec quels périphériques, ...De l'état de l'art du chapitre 3, nous avons choisi le service comme socle de base des interactions. Des périphériques vont interagir entre eux en appelant les services qu'ils proposent. De même qu'un humain va interagir avec un objet, dans notre cas un meuble intelligent tel une vitrine, via des services, comme un service de selection d'objets via une dalle tactile. Une fois ce choix établi, il nous faut maintenant une plate-forme technique, basée sur le paradigme SOA et capable d'accueillir des services provenant de fournisseurs de types et de technologies hétérogènes.
L'autre problématique à laquelle nous nous sommes attachées concernent l'industrialisation pour la production d'environnements intelligents. Nos cibles étant des entreprises différentes, nous avons alors des contraintes très fortes liées à l'intégration de nos solutions dans les différents SI. Mais nous devons aussi permettre une personnalisation forte de nos systèmes pour permettre à nos clients de s'approprier nos produits. Le coût de tels projets actuellement est top important. Il nous faut alors proposer une suite logiciel permettant la génération automatique de tout ou partie de nos solutions. Ces logiciels doivent être simples d'utilisation pour ne pas ête couteux en formation ou en interventions externes. Nous pouvons alors résumer cette problématique avec cette question : comment générer un ou des EVIs facilement et sans dépenser trop ? De là découlent des questions sur les méthodologies de développement de projet informatiques et sur la suite logicielle à mettre en place afin d'aider à la conception d'un environnement intelligent.

4.6 Solutions proposées

4.6.1 Solutions pour la plate-forme d'interactions

Pour répondre aux besoins liés à l'intelligence ambiante et aux contraintes fortes d'intégration, nous souhaitons implémenter les patrons d'intégrations définis dans ce chapitre. Mais il est également

nécessaire de permettre à un créateur d'EVI de définir des comportements entre différents services. Il faut alors pouvoir orchestrer des services au sein du système.

Dans le monde logiciel, les ESB [20] [Cha04] réunissent l'ensemble de ces patterns afin de fournir des architectures modulaires. Un ESB est un ensemble logiciel, de la couche middleware, facilitant l'intégration d'applications hétérogènes, comme le montre le schéma de la **figure 4.2**. Un ESB fournit au minimum l'ensemble de fonctionnalités suivantes :

- localisation transparente : un ESB permet de dissocier le consommateur d'un service du fournisseur de ce service
- conversion de protocole de transport : un ESB doit pouvoir intégrer des applications utilisant des protocoles de transports différents
- transformation de message : un ESB doit fournir la fonctionnalité de transformer un message d'un format vers un autre
- routage de message : un ESB doit pouvoir déterminer la destination d'un message entrant

On peut considérer l'ESB comme une nouvelle génération d'EAI (en français, Intégration d'applications d'entreprise) construite sur des standards comme XML, JMS ou encore les services web. Aussi, la différence majeure avec l'EAI réside dans le fait que l'ESB propose une intégration complètement distribuée grâce à l'utilisation des conteneurs de services. Ces serveurs contiennent la logique d'intégration et peuvent être déposés n'importe où sur le réseau. L'ESB possède quatre fondements :

- Le MOM [21] qui permet l'échange de messages de manière asynchrone. Ainsi chaque message est déposé sur une file d'attente avant d'être consommé par le destinataire.
- Les services Web qui permettent d'interfacer les applications avec le bus. Chaque service contient une logique d'intégration des messages (transformation, routage, etc.).
- Les transformations qui concernent les messages circulant sur le bus, elles sont essentielles dans un ESB car leur rôle est de permettre à des applications de converser même si elles définissent différemment leurs données.
- Le routage intelligent qui découple l'expéditeur du message de son destinataire. C'est en fait l'ESB qui va déduire la destination du message. Pour cela il se base sur le contenu du message et les règles qui ont été définies.

Peuvent venir se greffer à cette architecture des modules complémentaires :

- le Business Activity Monitoring (BAM) qui permet de suivre l'activité d'un processus métier
- le Business Process Management (BPM) qui a pour but de maîtriser l'orchestration des processus métier, c'est-à-dire l'enchaînement des échanges entre applications

Sur le marché, nous trouvons plusieurs solutions existantes. Les principales du monde openSource sont MULE[mul] et service Mix d'Apache [serb]. Les ESB actuels sont des composants lourds, qui nécessitent de gros serveurs d'applications. Pour nos besoins de système embarqué, il nous faut un ESB léger. Depuis 2007, des ESB dits *lightweight* sont apparus, comme le projet Apache Camel[cam]. Mais ces projets sont relativement récents et n'étaient pas opérationnels lors de nos travaux.

20. Enterprise Service Bus
21. Middleware Orienté Message

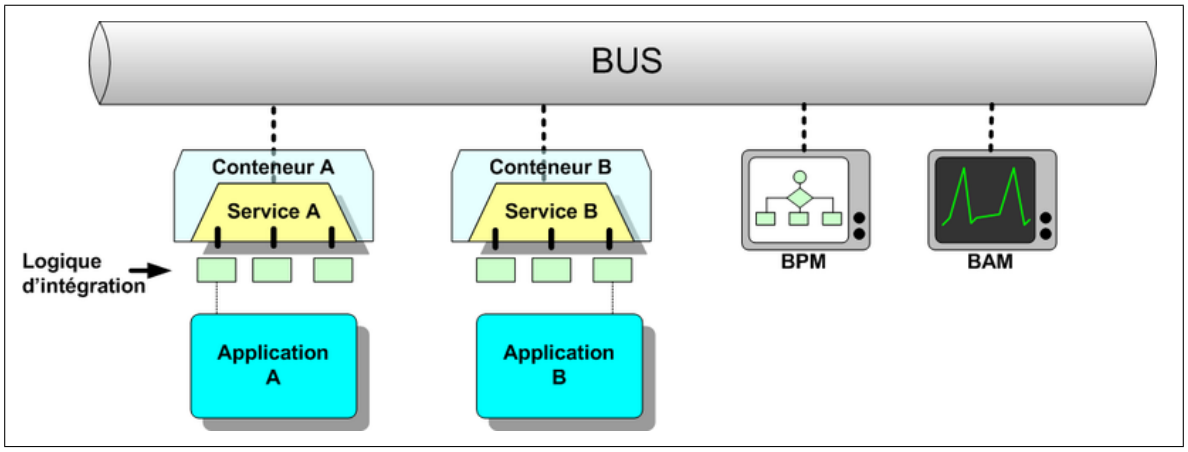

FIGURE 4.2 – Architecture d'un ESB

4.6.2 Solutions pour la création d'EVI

Pour répondre notamment aux problématiques de coûts lors de projets informatiques, les nouvelles méthodes de développement peuvent sembler être une bonne piste. Ces procédés de développement dits *agiles*, comme l'eXtreme Programming[Bec02] [BBMW02], permettent effectivement de réduire les coûts de développements par rapport à une méthodologie classique. Mais ces méthodes n'agissent qu'en amont du projet et ne résolvent pas notre problématique. Dans notre cas, il est important de pouvoir se définir un cadre de développement basé sur un socle réutilisable pour générer n'importe quel EVI et pour n'importe quel client. Nous avons pour cela regardé les travaux autour des patrons de conception permettant la réutilisation de code [BCO+02].

Après étude des différents patrons de conception existant [Gam95], nous avons commencé à travailler sur le patron *fabrique*, illustré dans la **figure 4.3**. Ce patron, issu de la programmation orientée objet, permet de créer des objets dont le type est dérivé d'un type abstrait. Le créateur de l'objet ne connait alors pas la classe exacte de l'objet mais seulement la sous-classe qui l'intéresse.

Plusieurs fabriques peuvent être regroupées en une fabrique dite *abstraite* ce qui permet d'instancier des objets dérivant de plusieurs types abstraits différents. Ce patron de conception est lui appelé *fabrique abstraite*. Le schéma UML de ce patron de conception est représenté dans la **figure 4.4**.

Une fabrique abstraite englobe un ensemble de fabriques. L'utilisateur de ce patron doit alors créer une implémentation de la fabrique abstraite puis il doit utiliser les interfaces génériques de la fabrique pour créer des objets concrets de la fabrique. L'utilisateur finale ne connait pas la fabrique qui a permis de générer l'objet. Ce patron de conception permet de séparer les détails d'implémentation d'un ensemble d'objets de leurs usages génériques. Une fabrique est un endroit du code où sont construits des objets. Le but de ce patron de conception est d'isoler la création des objets de leur utilisation. On peut ainsi ajouter de nouveaux objets dérivés sans modifier le code qui utilise l'objet de base. Avec ce patron de conception, on peut interchanger des classes concrètes sans changer le code qui les utilise, même à l'exécution.

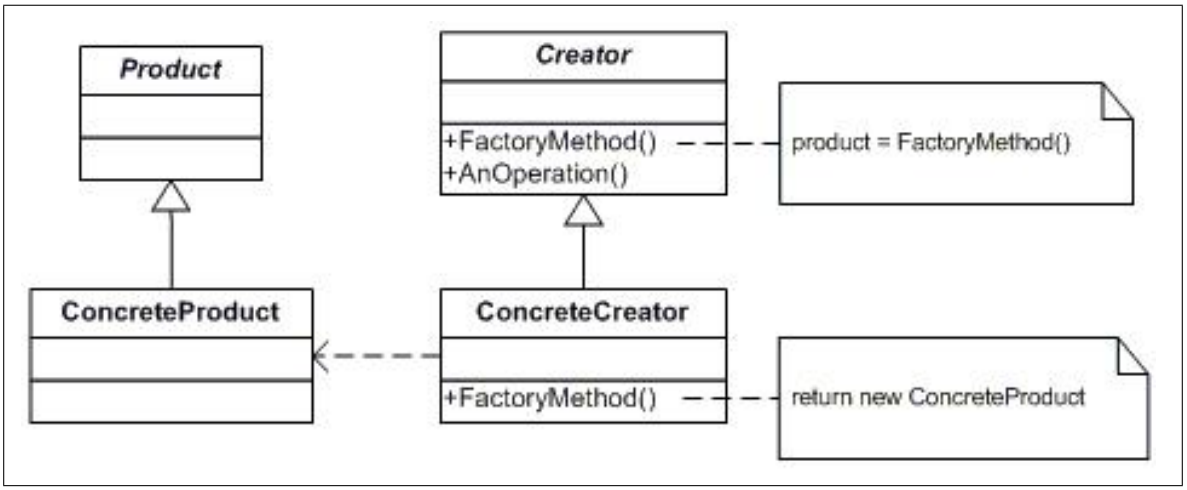

FIGURE 4.3 – Diagramme UML du pattern factory

Si l'on revient à la problématique liées à la création d'EVI, l'utilisation d'une fabrique abstraite répond à nos contraintes. La notion de fabrique logicielle (software factory) dérivée des patrons de conception précédents est la solution que nous avons alors retenue pour nos travaux.

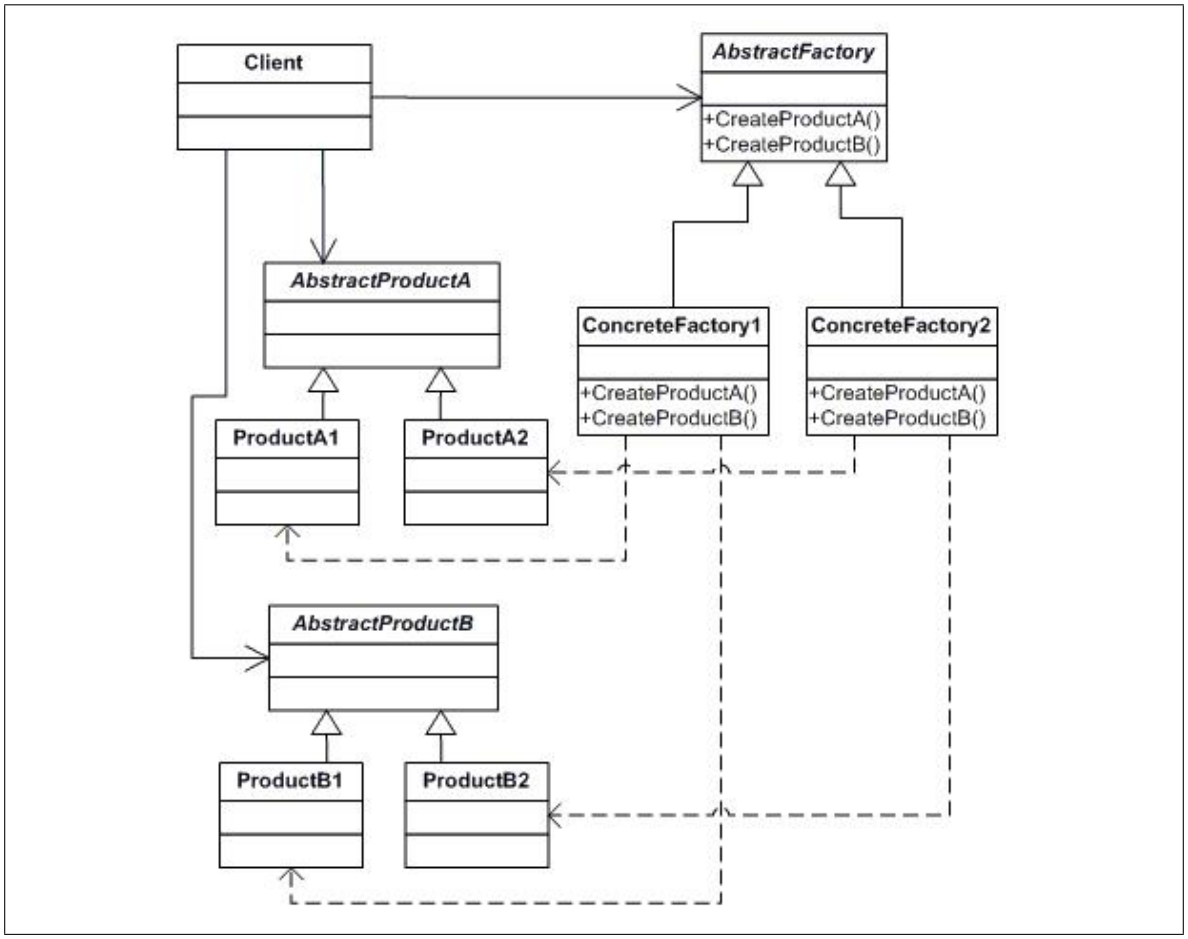

FIGURE 4.4 – Diagramme UML du pattern abstact factory

4.7 Spécifications

4.7.1 Fabrique logicielle

Le but d'une fabrique logicielle est de concevoir des solutions indépendantes. Une usine logicielle est constituée d'un ensemble de frameworks (avec une notion de composition et d'orchestration des services applicatifs), de librairies et d'outils dont certains sont fournis clés en main par l'Editeur (Microsoft, ...). Les outils majeurs dans une software factory comprennent notamment ceux qui permettent la gestion du cycle de vie d'un projet. Les projets deviennent pilotés avec par exemple des tableaux de bord.

Une usine logicielle est en réalité une structure organisationnelle qui se spécialise dans la production de logiciels informatiques ou de composants logiciels en fonction des spécificités définis par les besoins des utilisateurs finaux. Une usine logicielle se base sur des patrons de conception usitée en génie logiciel. Grâce à une usine logicielle, un logiciel peut être créé par assemblage de composants ou de services prédéfinis. Une application composite est le résultat final d'une usine de logiciel. Au-delà d'une meilleure intégration, ces outils industrialisent le cycle de développement en intégrant une dimension méthodologique et en automatisant la chaîne de production des applications. Cela en obligeant certains profils à effectuer des tâches dans un ordre donné ou en les contraignant à respecter des étapes de validation. Pour ajouter une classe dans la gestion de configuration, le développeur devra, par exemple, l'avoir préalablement testée ou validée auprès de son responsable technique. Pour être efficace, cette démarche impose cependant de formaliser les processus et le rôle de chaque individu au sein de l'usine.

Ces plates-formes viabilisent et accélèrent également les projets en automatisant la fin du cycle de développement. Le logiciel créé est assemblé, compilé, testé et publié par l'usine via des systèmes automatisés.

La notion de Software factory est expliquée plus en détails dans le chapitre 6.

4.7.2 OSGi

L'OSGI Alliance, créée en 199, est une société indépendante à but non lucratif. Elle comprend de nombreuses acteurs industriels (IBM, Microsoft, ...). Son but est de proposer un ensemble de spécifications qui offrent un environnement orienté services permettant de gérer le cycle de vie des applications. Cet environnement est basé sur Java, profitant ainsi de la possibilité de charger et/ou décharger dynamiquement des classes au cours de l'exécution d'une application. OSGi a été créée au départ pour des matériels embarqués, comme des passerelles réseaux ou résidentielles, dont les capacités mémoires et d'exécution sont limitées. De plus, ces systèmes ont des contraintes fortes en ce qui concerne le cycle de vie des composants logiciels : de nouveaux composants doivent pouvoir être ajoutés dynamiquement durant l'exécution sans interruption des services existants.

4.7.2.1 Une plate-forme de services locaux

OSGi permet le déploiement, l'exécution, l'administration et le retrait de services logiciels à distance. OSGi se décompose en deux couches distinctes : une physique qui règle les problèmes de dé-

ploiement (installation, désinstallation), et une couche logique pour la description des services et leur utilisation (assemblage, exécution, ...).

OSGi possède les notions relatives aux architectures orientées services. Le canevas propose donc un registre dans lequel les services sont enregistrés. Ces derniers sont référencés par un contrat de service, représenté par une interface Java, et un ensemble de propriétés (version, vendeur, ...) permettant des recherches de service via le protocole LDAP. [22]. La plate-forme repose sur un système événementiel permettant la notification des composants s'exécutant sur OSGi afin de réagir dynamiquement en fonction des actions effectuées sur la plate-forme.

4.7.2.2 Le bundle

Le bundle est l'unité de déploiement d'OSGi. Concrétement, il s'agit d'un fichier JAR Java. Il contient les informations nécessaires au déploiement comme indiquée dans la **figure 4.5** : services requis, services exposés, version ...Ces informations sont inscrites dans le fichier MANIFEST, standard dans Java. Ce JAR contient également les classes et les éventuelles bibliothèques utilisées, mais uniquement les bibliothèques n'étant pas définies en tant que dépendances (ie services OSGi requis). Les bundles permettent de mettre en oeuvre le concept de composant : un bundle peut être comparé à un composant. Un bundle est donc un composant simple à mettre en oeuvre. Cette caractéristique permet aux conteneurs OSGi d'être très légers, répondant ainsi à la contrainte initiale, c'est à dire l'utilisation dans des systèmes embarqués.

Les conteneurs possèdent diverses caractéristiques permettant la gestion des bundles. Tout d'abord, ils ont la particularité de permettre la gestion des dépendances entre bundles. En effet, comme nous l'avons vu précédemment, un bundle s'appuie la plupart du temps sur d'autres afin de réaliser ses traitements. Un conteneur OSGi offre la possibilité de rendre visible dans le conteneur les packages de composants de manière très fine. La gestion de dépendances vis à vis d'une version d'un bundle est également supportée.

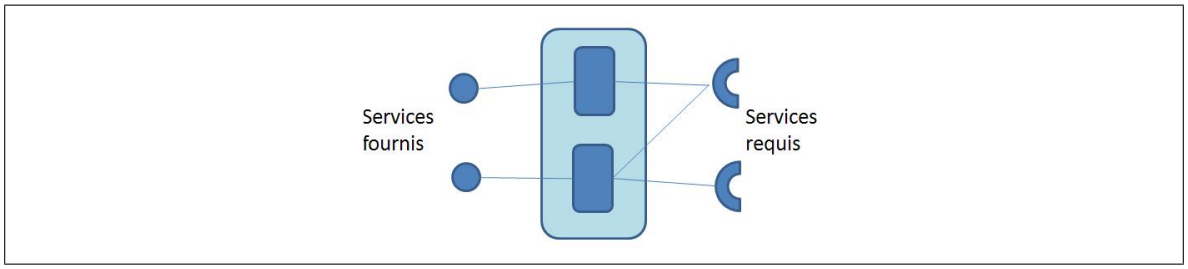

FIGURE 4.5 – Représentation logique des services d'un bundle

4.7.2.3 Implémentations

Il existe plusieurs plates-formes implémentant la spécification OSGi.

22. Lightweight Directory Access Protocol

Felix/Oscar Felix est l'implémentation Apache. Felix est issu d'un projet étudiant nommé OSCAR. Il a été repris la communauté Apache et rebaptisé Felix. Le framework Felix est disponible vierge de tout bundle. Il est alors possible de récupérer des bundles existants sur un repository distant, appelé OBR [23]. La communauté Apache fournit de nombreux bundles fonctionnels permettant ainsi de personnaliser nos propres plates-formes.

Felix/Karaf Karaf est un environnement basé sur Felix mais arrangé pour prendre en compte le framework Apache CAMEL. Nous l'indiquons ici car il s'agit du framework que nous avons utilisé lors de nos tests d'intégraton avec CAMEL.

Equinox Equinox est le framework OSGi du projet Eclipse. Il permet ainsi à Eclipse d'être un IDE modulaire par le système de plug-in. Un plug-in Eclipse est en réalité un bundle OSGi qui utilise les services fournis par la plate-forme Equinox de l'IDE et qui expose ses propres services.

Knopflerfish Au départ payant, Knopflerfish est un framework OSGi largement répandu. Ce framework a la particularité d'être fourni avec un ensemble de bundles fonctionnels, comme une interface graphique ou un bundle de logs.
Notre choix s'est porté sur les plates-formes Knoplerfish et Felix. Nous avons utilisé la première lors de nos développements. Le package initial fourni avec plusieurs services permet de pouvoir rapidement développer. Nous avons ensuite utilisé Felix lors de nos expérimentations car elle est plus légère en termes de place et de consommation mémoire.

4.7.3 UPnP

UPnP permet la création de réseaux autonomes spontanés. En utilisant l'auto-configuration des périphériques du réseau, UPnP permet l'arrivée et le départ du réseau, découvrir les dispositifs (*devices*) et les services associés de manière automatique en ne demandant que très peu voire pas du tout d'intervention humaine.

Ce protocole a été crée en 1999 par l'UPnP Forum réunissant une vingtaine d'entreprises, dont Microsoft et Intel. Leur but était d'inverser la tendance demandant de plus en plus de compétences aux utilisateurs pour configurer leurs équipements communicants. Les équipements doivent faire le travail seul car tout le monde ne dispose pas d'un administrateur système et réseau sous la main. L'architecture UPnP est alors conçu pour interconnecter dynamiquement tous les dispositifs disponibles sur un réseau local/domestique (PC, TV, box, console de jeu, cafetière, ...). Pour cela, UPnP s'appuie sur des protocoles standards, qui sont enter autres, IP (adressage des dispositifs), TCP et UDP, HTTP, XML. L'ensemble de la pile UPnP est décrite dans la **figure 4.6**. De cette manière, UPnP est complètement

23. Oscar Bundle Repository

indépendant de l'architecture matérielle, du système d'exploitation et des langages de programmations utilisés.

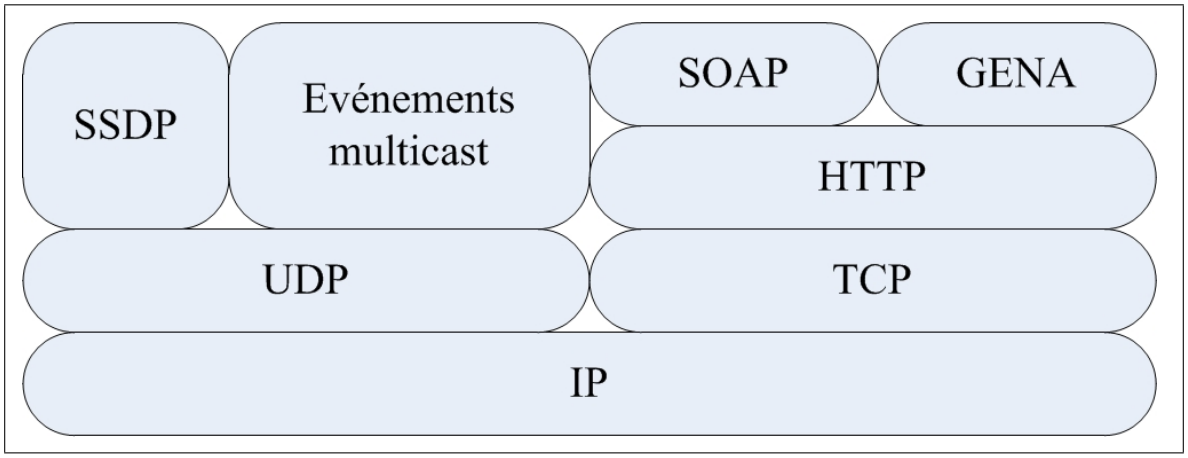

FIGURE 4.6 – Pile protocolaire UPnP

4.7.3.1 *Terminologie*

- *Device* : dispositif connecté sur le réseau qui implémente les protocoles nécessaires pour l'architecture UPnP. Possibilité d'avoir plusieurs *devices* dans un *device*. Dans ce cas là, il y a une hiérarchie de *devices*
- *Service* : unité de fonctionnalités implémentée par un *device*. Chaque service à un ensemble de méthodes ou action(un service est équivalent à l'interface d'un composant logiciel). Un service maintient un état cohérent via des variables d'états (équivalent aux variables membre dans une classe). Les changements sont notifiés de manière événementielle aux *control points* intéressés
- *Control point* : entité sur le réseau qui utilise les services d'un *device*

Nous allons maintenant présenter brièvement les différentes couches nécessaires à un environnement UPnP.

4.7.3.2 *L'adressage*

Dans un réseau UPnP, l'adressage se fait via des adresses IP. Chaque *device* doit implémenter un client DHCP et se connecter à un serveur DHCP à la première connexion. Si un aucun serveur n'est accessible, le *device* doit s'assigner lui-même une adresse IP (protocole IPv4LL - [CAG05]). Mais le *device* peut également utiliser un nom de domaine si il en obtient un via un serveur DNS.

4.7.3.3 *La découverte des services*

Le protocole utilisé pour la découverte dans un réseau UPnP est le Simple Service Discovery Protocol (SSDP). SSDP permet deux choses :

1. il permet à un *device*, qui vient d'arriver sur le réseau, d'envoyer aux *control points* les services qu'il propose.
2. il permet à un *control point*, qui vient d'arriver sur le réseau, de chercher des *devices* sur le réseau.

Les messages échangés sont très courts, même le message concernant la description d'un *device* (ou de l'un de ses serices) qui n'est constitué principalement que d'informations essentielles comme par exemple le type du *device*, son identifiant et un lien vers d'autres informations détaillées.

4.7.3.4 *Description des périphériques*

Une fois qu'un point de contrôle a découvert un *device*, nous avons vu via le protocole SSDP qu'il ne connaît pas encore assez d'informations sur le device pour pouvoir l'utiliser. Le point de contrôle peut alors retrouver la description détaillée du *device* à partir du lien (une URL) fourni dans le message SSDP. La description UPnP d'un *device* est au format XML est contient des informations telles que : the model name and number, serial number, manufacturer name, URLs to vendor-specific web sites, etc. Nous trouvons également dans la description une liste de *devices* embarqués ou de services et également des URls utilisées pour le contrôle du *device*, pour les évènements et pour la présentation. Pour chaque service, la description fournit un ensemble de commandes, ou actions, auxquelles répond le *device* et des paramètres, ou arguments, pour chaque action. La description d'un service fournit également une liste de variables, dites *variables d'état*. Ces variables représentent l'état du service, donc du *device*, au cours de l'exécution.

4.7.3.5 *Control*

Une fois la description d'un *device* reçue, le point de contrôle peut envoyer des actions via le service d'un *device*. C'est la phase dite de contrôle. Un message de contrôle est envoyé au *device* à l'URL de contrôle associée au service (fournie dans la description). Ces messages sont également des messages au format XML, plus particulièrement des messages SOAP. Le service peut alors retourner une ou plusieurs valeurs, comme un appel de méthode classique. Les effets de bord de l'action, s'il y en a (allumage d'une ampoule par exemple), sont modélisés par les *variables d'état*.

4.7.3.6 *Gestion des évènements*

Une des fonctionnalités intéressantes d'un réseau UPnP est la possibilité de gérer des évènements, des notifications. Le protocole dédié dans UPnP est le protocole GENA, pour General Event Notification Architecture. Comme vu dans la partie 4.7.3.4, la description UPnP d'un service fournit une liste d'actions auxquelles le service répond mais aussi une liste de *variables d'état* permettant de connaître l'état d'un service en temps réel. UPnP permet au service de publier sur le réseau des mises à jour lorsque ces variables sont modifiées. Un point de contrôle peut alors souscrire au service pour recevoir en temps réel ces mises à jour. Ces mises à jour sont des messages évènementiels. Ces messages, également au format XML, contiennent le nom des variables d'états modifiées et les nouvelles valeurs de

ces variables. Un point de contrôle reçoit un message évènementiel particulier lorsqu'il souscrit à un service : ce message contient les noms et valeurs de toutes les variables d'état pour permettre au point de contrôle d'initialiser l'état du service. Ce support évènementiel permet à tous les points de contrôle d'un *device* de rester informer des effets de toutes les actions reçues par le *device*. En pratique, tous les points de contrôle qui ont souscrits à un service reçoivent les notifications de toutes variables d'états dont la valeur a été modifiée, peu importe la raison qui a entrainé le changement de valeur.

4.7.3.7 *Presentation*

Enfin, UPnP fournit une couche optionnelle appelée "Présentation". Un *device* peut fournir une URL de présentation. Cette URL pointe sur une page à partir de laquelle un point de contrôle peut contrôler le device et/ou consulter l'état de ses services (via les variables d'état). La page de présentation ne répond à aucune norme (mise à part qu'elle doit s'ouvrir dans un navigateur) ni contraintes en terme de contenu ou d'actions proposées.

Cette présentation d'UPnP met en lumière que cette technologie permet à un utilisateur d'être en interaction avec son environnement informatique, non seulement par la possibilité d'utiliser des périphériques présents autour de lui, mais également en lui apportant des éléments de contexte via les variables d'états des services.

4.8 Synthèse

En recoupant l'ensemble de nos solutions répondant aux problématiques posées, nous pouvons commencer à entrevoir la solution que nous envisageons. D'un point de vue technique, la plate-forme d'interactions doit se baser sur le concept d'ESB et les technologies UPnP/OSGi. D'un point de vue de la conception, le patron de conception *fabrique* semble répondre à nos besoins. Enfin, il est nécessaire de fournir avec notre solution une suite logicielle facilitant la génération d'EVI et l'utilisation de la plate-forme d'interactions.

Le résultat de nos travaux est une usine logicielle permettent de générer facilement des EVIs pour différents clients. Grâce à cette usine, il est facile de manipuler des meubles augmentés en les considérant comme des fournisseurs de services. L'industrialisation est assurée par le cadre offert par l'usine logicielle. Ces services peuvent être exécutés sur une plate-forme logicielle basée sur les technologies OSGi et UPnP afin de répondre aux problématiques de l'intelligence ambiante.

Dans nos travaux, nous avons voulu mettre en avant l'importance de la couche de communication pour les projets dits *pervasifs*.

Chapitre 5

Infrastructure de communication

Dans nos travaux, nous avons voulu mettre en avant l'importance de la couche de communication pour les projets dits *pervasifs*. La première brique implémentée de notre usine logicielle a donc logiquement été l'infrastructure de communication. Cette infrastructure, appelée YMCA, est basée sur le concept d'ESB présenté au chapitre 4. Cet ESB, en plus des services de base offert par ce genre d'intergiciel, permet la communication entre périphériques non-interoperables à l'origine. Nous nous sommes pour cela inspirés du concept de micro-monde technologique et de ponts entre ces micro-mondes pour réaliser les échanges. Ce chapitre permet de détailler le fonctionnement de YMCA.

5.1 YMCA : un ESB dédié aux EVI

Notre usine logicielle s'appuie sur une plate-forme intergicielle, YMCA, que nous avons développée spécifiquement afin de répondre aux besoins liés à la flexibilité pour l'intégration. Afin d'illustrer au mieux le fonctionnement de notre plate-forme, nous avons choisi de suivre le scénario suivant :

- nous sommes dans un rayon de supermarché, l'utilisateur est un vendeur en présence d'un client. Il souhaite effectuer le déport d'un comparatif de trois GPS sur un écran
- le rayon dispose d'un écran LCD permettant d'afficher des informations
- le vendeur possède un appareil mobile communiquant sur le réseau
- on suppose l'existence d'un SI permettant de récupérer toutes les données nécessaires

Si on résume en termes d'infrastructure technique et de dispositifs, nous avons donc :

- un équipement mobile (type UMPC). Le vendeur dispose d'une application communiquant en JMS
- un écran LCD compatible UPnP v1
- un système d'informations, propre à l'entreprise, permettant de récupérer des informations propres à un produit (fiche produit), de récupérer un comparatif de produits
- un service de déport développé en Java, communiquant avec l'intergiciel via JMS

Pour correspondre à ce scénario, nous avons alors défini une configuration type de l'environnement EVI visé. Cette configuration est décrite dans la **figure 5.1**.

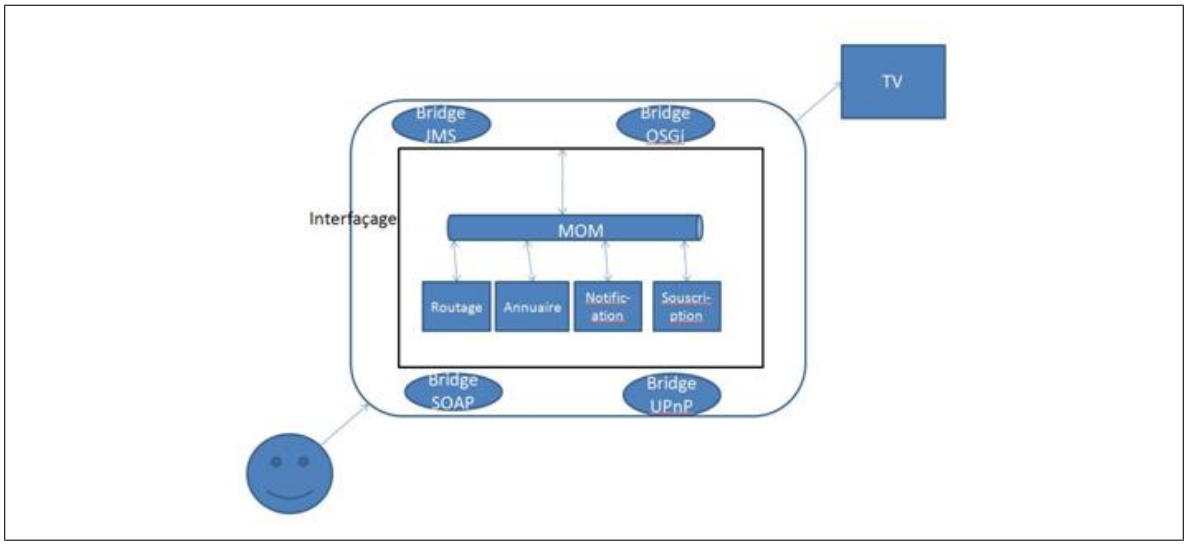

FIGURE 5.1 – Architecture logique du scénario

5.1.1 YMCA : constitution de l'ESB

Le scénario précédent permet de définir la fonctionnalité principale de YMCA : permettre à un usager le contrôle d'un dispositif, via les services que celui-ci propose, sans connaître ni la technologie ni le protocole utilisés par ce dispositif. Au sein de YMCA, un dispositif est alors vu comme un fournisseur de services au sens SOA. Dans un contexte d'utilisation réelle, l'homogénéité des dispositifs présents dans un rayon n'est pas assurée, tout dépend du marché. Les dispositifs peuvent provenir de constructeurs différents, respecter des protocoles particuliers, standards ou propriétaires et répondre à des normes spécifiques. De plus, les différents constructeurs n'utilisent pas forcément les même protocoles/piles de communications, ils n'ont pas non plus de spécifications communes (terminologie différentes, technologies différentes). Le problème revient donc à permettre à un client d'invoquer un ou plusieurs services, sans connaître les spécifications techniques de ce dernier. La problématique d'intégration est donc ici omniprésente. De plus, un client doit pouvoir être informé immédiatement du changement d'état d'un appareil, soit par demande explicite, soit par propagation d'évènement. YMCA, en répondant à ces problématiques, apparaît ainsi comme un E.S.B. autorisant l'interconnexion de technologiques hétérogènes.

Notre ESB se base sur 4 éléments clés de la définition d'un ESB [RD08], comme on peut le voir dans la **figure 5.2** :

- Un middleware orienté messages qui permet l'échange de messages de manière asynchrone.
- L'interface, représenté par des ponts (*bridges*) qui permettent d'interfacer les applications avec le bus. Ils permettent la réception et le routage des messages de protocoles/piles de communication différents.
- Les transformations des messages circulant sur le bus. Elles permettent l'interopérabilité entre différents protocoles/piles de communication. Elles sont assurées dans notre cas par les bridges.

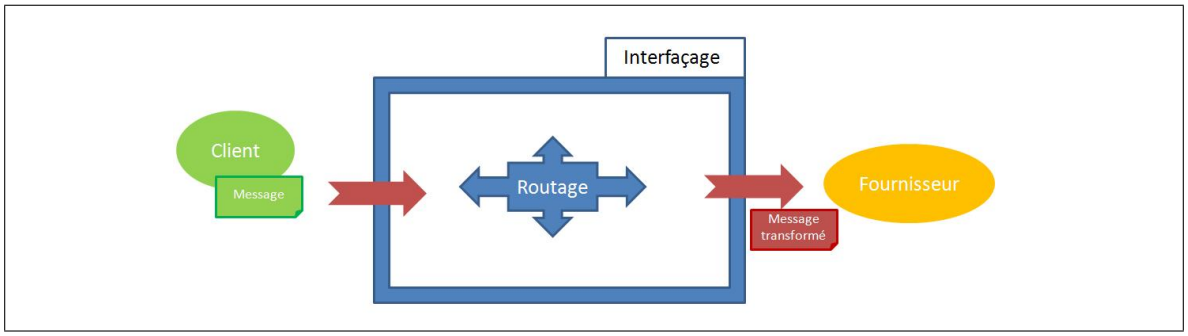

FIGURE 5.2 – Vue logique globale de YMCA

– Le routage qui découple l'expéditeur du message de son destinataire. C'est en fait l'ESB qui va déduire la destination du message.

Nous avons choisi d'implémenter notre propre ESB pour respecter les contraintes suivantes :

– Contrainte forte d'indépendance (plate-forme, système d'exploitation) pour répondre aux besoins de différents clients
– Besoin d'un dispositif léger, pouvant fonctionner sur un système embarqué

Les ESBs présents sur le marché sont relativement lourds à mettre en place et nécessitent des machines possédants des ressources de calcul importantes (transformations, conteneur d'applications). De plus, ils ne sont pas compatibles avec les principaux langages existants, car il n'existe pas de mécanisme d'adaptation standard. Pour toutes ces raisons, nous avons décidé d'implémenter un « ESB lightweight ». Contrairement à un ESB classique, un lightweight ESB se définit par :

– Une grande légèreté : pas de conteneurs, d'APIs sophistiquées de connecteurs, de structures de données complexes mais de simples messages composés d'un corps et d'une entête.
– L'éloignement des standards (Java Business Integration - JBI et Service Component Architecture - SCA) pour se cantonner à des types simples. Ces plates-formes utilisent massivement les API Java EE pour les connecteurs et les transactions (JDBC, JMS, JCA, JTA, etc).

D'un point de vue technique, YMCA est un intergiciel orienté messages. YMCA répond à 3 grandes fonctionnalités caractéristiques de ce type d'intergiciels :

– La découverte de services
– La description de services
– Le service de Messaging

YMCA peut aussi être assimilé à un ESB léger ou encore intergiciel d'intégration. Le terme « ESB léger » est utilisé [liga], [ligb] pour désigner des ESBs utilisant les POJOs et les API Java EE . En plus de ces caractéristiques, la plate-forme ainsi développée au cours de nos recherches utilise l'architecture OSGi (cf partie précédente) pour permettre une grande flexibilité.

5.1.2 YMCA : un intergiciel pour l'informatique pervasive

En plus des fonctionnalités principales, YMCA propose des services visant à faciliter le développement d'applications *pervasives* :

- la souscription à un service : un client peut souscrire à un service et sera notifié dès que le service deviendra disponible (consécutivement à la demande si le service est déjà présent). Sur le départ d'un service, une notification est également envoyée aux souscripteurs pour les prévenir de l'indisponibilité du service.
- la notification sur une variable d'état : les services ont la possibilité d'exposer des variables d'état. Un client peut alors demander à être notifié suite à un changement de valeur d'une variable d'état associée à un service et ainsi en récupérer la nouvelle valeur.

5.1.3 YMCA et les EVI

La spécialisation de notre plate-forme pour l'informatique persavise permet son utilisation pour l'implémentation d'un Espace de Vente Intelligent, selon la définition que nous en avons fait. En reprenant les éléments du scénario de base, nous pouvons alors considérer l'UMPC du vendeur comme un client. Le SI et l'écran LCD sont alors vus comme des fournisseurs de services :

- le SI fournit des services de récupération de fiches produits
- l'écran fournit un service d'affichage d'informations

Les trois dispositifs sont interconnectés via YMCA. Ainsi pour reprendre le scénario de base, via l'utilisation de YMCA, l'UMPC du vendeur peut se connecter au SI du magasin, récupérer les informations dont il a besoin puis utiliser le service offert par l'écran LCD pour lui faire afficher un comparatif.

Nous allons maintenant présenter en détail comment peut se former une telle communication entre dispositifs hétérogènes, en commençant par définir la notion de service au sein de YMCA.

5.2 La notion de service au sein de YMCA

Un EVI est selon notre point de vue un ensemble de dispositifs proposant des services, soit à des utilisateurs, soit à d'autres EVIs, soit à des applications, ...D'un point de vue génie logiciel, un service ici peut être vu comme une méthode proposée par un objet. Afin de pouvoir être appelé et utilisé par n'importe quel autre dispositif, le service doit pouvoir être décrit dans un format compréhensible par YMCA. Nous avons alors défini le format présenté ci-après.

5.2.1 Description de services

Les services à l'intérieur de l'intergiciel sont décrits en XML. Le but principal de YMCA est de faire dialoguer des périphériques, soit entre eux, soit avec des clients. Nous nous sommes alors basés sur la description fournie par les spécifications UPnP (cf Service-Template). Les principales caractéristiques pour décrire un service sont :

- `Fournisseur` : Un fournisseur est une entité qui fournit un ensemble de services. Il est caractérisé par un nom unique, qui lui sert d'identifiant, un type, index permettant d'inférer dessus, et par un protocole (pile, protocole), afin de savoir comment on peut dialoguer avec, équivalent du « binding » dans wsdl. Le type est important car il va déterminer les services que proposera le fournisseur. Par exemple, un fournisseur de type « Printer » fournira des services relatifs à l'impression.
- `Service` : Il consiste en une fonction ou fonctionnalité (vue comme un ensemble de fonctions) proposé par un dispositif. Il est divisé en actions qui constituent autant d'opérations spécifiques que le service peut réaliser. Le service est caractérisé par son nom et par une liste d'opérations, appelées ici action. De plus, un service possède un ensemble d'attributs nommés variables d'états (state var). Ces variables permettent de représenter l'état d'un dispositif à tout moment.
- `stateVar` : une stateVar est une variable décrivant l'état du dispositif pendant son exécution. Une stateVar peut être attachée à plusieurs services proposés par le dispositif. Ainsi à tout moment, on peut connaître l'état d'un appareil. On peut ainsi connaître le flux vidéo diffusé par un écran par exemple, ou savoir si ce même écran est disponible ou non.
- `Action` : une action correspond à une opération que peut réaliser le service
- `Methode` : une méthode est l'implémentation d'une action. Elle permet de faire le lien entre le nom d'une action et sa référence (instanciation).
- `Param` : paramètre d'entrée d'une méthode
- `Retour` : retour d'une méthode. Le retour est caractérisé par un type. Actuellement, un retour peut prendre les types définies dans le tableau suivant. Nous avons réalisé l'intersection des types primitifs des langages objets courants afin de respecter une plus grande hétérogénéité.

Nous pouvons remarquer que l'élément de plus haut niveau dans la description n'est pas le service, mais le fournisseur de services, celui qui va proposer le service. Pour accéder à la description d'un service, il faut alors connaître au préalable son fournisseur ou au moins le type de ce fournisseur. Un fournisseur d'un type et d'un protocole donnés propose un ensemble de services prédéfinis. Un service expose des variables d'état et des actions. Voici un exemple tiré du scénario (le schéma est disponible en annexe) :

```
1  <fournisseur nom "Ecran Samsung" protocole "UPnP" type "htmlRenderer"
2                  xsi "http://www.w3.org/2001/XMLSchema-instance"
3                  noNamespaceSchemaLocation "Service.xsd">
4      <service name "Affichage">
5          <statevar>
6              <var name "url" />
7              <var name "dispo" />
8          </statevar>
9          <actions>
10             <action name "AfficheHTML">
11                 <methode name "displayHTML" />
12                 <param name "url" type "string" />
```

```
13                  <retour type "void" />
14              </action></actions>
15      </service>
16 </fournisseur>
17
```

L'écran LCD utilisé dans le scénario est un fournisseur de service de type "htmlRenderer". Ce type indique qu'il est capable de fournir des services pour afficher du contenu au format HTML. Ce fournisseur propose ici un service *Affichage*. Ce service possède deux variables d'état :

- url : permet d'indiquer l'url de la page affichée
- dispo : permet d'indiquer si l'écran est disponible ou non

Grâce à ces variables, un second vendeur peut savoir quelle est l'url affichée à l'écran et aussi savoir si l'écran est disponible ou non.

Le service est lui-même décomposé en *actions*. Une action donne des indications techniques sur la façon d'appeler le service. Dans notre cas, nous savons qu'il faut appeler une méthode appelée *AfficheHTLM* avec un paramètre d'entrée *url* de type chaîne de caractères. Une action indique également le type de retour du service appelé. Dans notre cas, nous avons utilisé le mot clé « void » permettant d'indiquer que la méthode ne renvoie pas de retour. Nous avons surchargé cette description avec des méta-données, permettant d'inférer sur les services. Dans [PNGL+09], nous proposons un ensemble de méta données afin de décrire le plus finement possible des services. Cette description est importante car c'est elle qui conditionne ensuite les recherches de services. Le tableau 5.1 donne une vue d'ensemble de notre proposition. Nous avons classé ces méta données par catégories.

Une fois le format défini, YMCA propose des fonctionnalités permettant aux différents utilisateurs de les trouver et de les utiliser. Le service *annuaire* de YMCA est le service de gestion de tous les autres services, permettant notamment la découverte dynamique des services qui s'inscrivent sur l'intergiciel.

5.2.2 Découverte de services : le service annuaire

YMCA permet la découverte de services pendant l'exécution. La plate-forme propose en effet un service d'annuaire de services qui s'occupe d'enregistrer les services et permet ainsi aux consommateurs de les retrouver. Cet annuaire contient l'ensemble des services disponibles et leur description au format XML comme défini dans la partie précédente, tant au niveau technique qu'au niveau sémantique. Il fournit aussi une interface permettant d'interroger l'annuaire. Il existe deux façons de découvrir les services :

- Synchrone : l'annuaire propose une API pour interroger l'annuaire. On peut ainsi découvrir les services correspondant à un type particulier, récupérer des informations relatives à un service (localisation, fournisseur, ...).
- Asynchrone : un autre service du middleware, le service souscription, permet de s'inscrire à un service de notification d'apparition de services. Un client peut ainsi demander à souscrire à un type de service. Lorsqu'un service du type demandé apparait, le client reçoit une notification.

catégorie	**méta données**
General	name, description, language, owner, type, entityType
Meta-metadata	metadataCreator, metadataValidator, creationDate, validationDate, language,format
Life-cycle	creator, dateCreated, version, status, contributor, publisher, dateUpdated, extentOfValidity
Right	IP, accessRight, signature, provenance, dateCreated, dateUpdated
Technical	URI, resource, resourceURI, resourceFormat, replacedBy, realisation, modeOfInteraction
ServiceRequirement	input(name, type, value, ontologyURI), output(name, type, value, ontologyURI),expectedEffect
DomainContent	listDomainConcepts
Context	roleModels, location (coordinate, spatialLocation, locationRelativity), physical (deliveryChannel, deliverySystem, deviceModel, tool), informaticResource(hardware, software), temporal (temporalCoverage, frequencyRequirement)
Quality	qualityRating, trustRating, qualityGuarantee, networkedQoS, accuracy, performance, reliability, robustness, scalability, security, availability, stability)

TABLE 5.1 – Meta-données associées aux services

Cet annuaire fournit aussi une interface de gestion des services, permettant entre autre d'ajouter ou supprimer des services. Ainsi, si un service apparait, il est ajouté à l'annuaire, soit dynamiquement, soit après que le service ait prévenu de son arrivée. Cela dépend du protocole utilisé par le service. Contrairement à des services d'annuaire comme celui de CORBA, nous ne faisons pas qu'un simple lien entre un nom de service et l'implémentation de celui-ci, nous proposons une description complète du service, dont chaque caractéristique sert d'index de recherche. Nous avons redéveloppé notre propre service d'annuaire pour le faire correspondre à notre description de services. Cela était possible car la complexité d'un EVI est telle que le nombre de services à gérer n'est jamais élevé.

La gestion des services est donc assurée par l'annuaire. Pour terminer la présentation de YMCA, il reste encore à détailler la communication entre les différents services.

5.3 La communication entre services

5.3.1 La gestion des messages

Comme décrit précédemment, YMCA permet la communication entre des fournisseurs et des clients. Cette communication est basée sur des échanges de messages. L'échange de messages se fait principalement de façon asynchrone. Les messages sont déposés sur des files de message jusqu'à ce qu'ils soient consommés par le destinataire. Le format des messages est le suivant :

Message = Entête , corps de message ; Entête = Commande,idClient ; Corps de message = "|",terme ;

Quand un client envoie un message pour un fournisseur, YMCA récupère le message via la couche d'interfaçage. Le service de routage se charge de transmettre le message. Étant basé sur OSGi, nous avons choisi d'utiliser le service EventAdmin introduit depuis la version R3 de la spécification OSGi. Ce service offre un modèle de communication événementielle entre les bundles par publication/souscription. L'objet événement (Event) est associé à un sujet (topic). Dans notre cas, chaque bundle possède son propre Topic. L'éditeur poste (i.e. publie) un événement au service EventAdmin qui le diffuse (potentiellement en parallèle) à tous les souscripteurs du sujet.

Ces messages sont génériques et sont identiques pour tous les périphériques du système. Des composants particuliers, appelés bridges, permettent la communication entre tous les acteurs d'un EVI.

5.3.2 Communication hétérogène : le rôle des bridges

L'intérêt d'un ESB est de pouvoir interconnecter des applications, services, clients, ... de technologies différentes sans contrainte pour le client. Le but étant de créer un réseau de services interopérables. Nous introduisons ici le terme de micro-monde. Nous reprenons la définition de micro-monde donné dans les travaux de David Menga [osgi/upnp] : un micro-monde correspond à réseau de dispositifs partageant le même protocole ou pile de communication. Tous les équipements n'utilisent pas les technologies, protocoles de communication, ...Chaque réseau de composants homogènes crée ce qu'on appelle un « micro-monde ». Il s'agit d'un sous-ensemble de l'environnement pervasif dans lequel nous nous trouvons. Chaque sous-ensemble est déterminé par la technologie employée. Dans notre projet, nous avons considéré les « micro-mondes » suivants :

- OSGi
- JMS
- UPnP
- SOAP (Web services)

Pour chacun de ces « micro-mondes », nous avons implémenté un connecteur permettant à un service de s'interfacer avec YMCA. Nous avons appelé ce connecteur bridge. Ces bridges servent donc de passerelle entre le middleware et le monde extérieur. Ils doivent donc transformer les messages qui leur parviennent. Les règles de transformation diffèrent selon le micro-monde.

5.3.2.1 Les bridges : transformation de messages

Le principe est toujours de récupérer un message issu du protocole du micro-monde et de le transformer pour le faire correspondre à un message du middleware, et vice versa. Les messages internes au middleware sont des events diffusés par le service Event Admin de l'implémentation OSGi. Le bridge doit donc récupérer le message issu du micro-monde et le transformer en event. Un pont entre l'EventAdmin et un micro-monde particulier ne requiert pas de modification dans le code du service implémentant l'EventAdmin : un pont est un émetteur/récepteur comme un autre qui capture tous les événements reçus par l'EventAdmin. Le pont convertit les événements de l'EventAdmin pour le micro-monde et les propage. De la même façon, un message reçu via le Micro-monde est converti puis envoyé vers l'EventAdmin. Lorsque l'on implémente un connecteur, il faut donc lever le problème suivant : comment sont convertis les événements de l'EventAdmin en messages pour le Micro-monde et vice-versa. Le principe d'un bridge est résumé dans la **figure 5.3** qui représente l'architecture logique de cet élément.

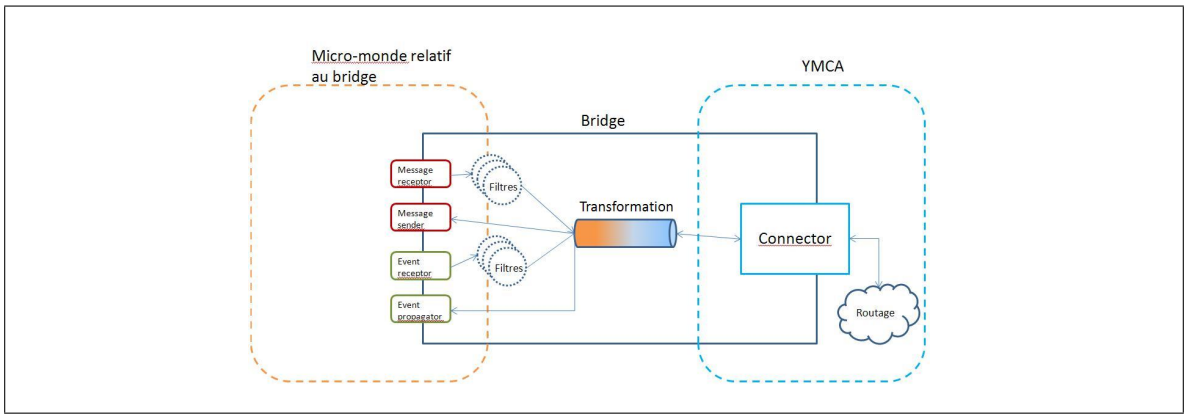

FIGURE 5.3 – Architecture d'un élément Bridge

Voici un exemple de transformation : un écran UPnP permet d'afficher des pages HTML. Un client JMS décide de l'invoquer en passant comme argument une url valant : `http://localhost/comparatif.swf`.

Le message JMS issu du client est alors :

```
INVOKERET|"+queueClient+"|Samsung|affichage|AfficherHTML|http://localhost/comparatif.swf
```

Le bridge JMS récupère le message, le transforme en event et l'envoie dans le middleware. L'event résultant est le suivant :

```
INVOKERET|clientJMS|EcranGPS|affichage|AfficherHTML|http://localhost/comparatif.swf
```

Grâce à l'annuaire, le service routage sait que ce fournisseur est de type UPnP et donc transfère l'event sur le connecteur UPnP. Ce connecteur récupère le message et reconstruit le message UPnP

correspondant pour l'envoyer au périphérique. Le connecteur agit comme un point de contrôle dans son micro-monde. Le message UPnP résultant est :

```
1  POST /service/power/control HTTP/1.1
2  Content-Type: text/xml; charset="utf-8"
3  HOST: url:port
4  Content-Length: 287
5  SOAPACTION: " urn:schemas-upnp-org:device:htmlrenderer:1#AfficherHTML"
6  Connection: close
7  <?xml version="1.0" encoding="utf-8"?>
8  <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" s:encodingStyle="http://schem
9  <s:Body>
10 <u:AfficherHTML xmlns:u="urn:schemas-upnp-org:service:affichage:1">
11 <Url>http://localhost/comparatif.swf</Url>
12 </u:AfficherHTML>
13 </s:Body>
14 </s:Envelope>
15
```

Grâce à l'annuaire, le connecteur UPnP parvient à reconstruire le message en récupérant les données dont il a besoin.

5.3.2.2 Les bridges : les services offerts

Dans notre architecture, chaque bridge, tel que défini précédemment, est implémenté par un bundle permettant le lien entre le « framework » OSGI et une entité externe (fournisseur de services ou client) de protocole différent (UPNP, JMS, WS, IVY, ...).

Un bridge doit donc remplir les rôles suivants :

1. Inscription d'un service dans l'annuaire du middleware :
   - « Découverte » du service dans le micro-monde associé au pont :
     - dynamiquement par la récupération d'une notification (annonce du service au sein de son micro-monde) ;
     - dynamiquement par la déclaration explicite du service auprès du pont ;
     - statiquement via un fichier de description de services associés au pont
   - Inscription du service dans le bundle annuaire :
     - Soit dynamiquement à l'aide de filtres spécifiés dans des fichiers « bridges.properties »
     - Soit demandé explicitement par le service lors de sa découverte statique via un fichier de description
2. Invocation d'un service par un client du micro-monde et gestion du retour :
   - Le client envoie une invocation au pont,
   - Le pont récupère le message, lui ajoute le protocole du client et le transmet au routage.
   - Si il y a un retour attendu :

  - Le pont récupère ensuite la notification de retour du service,
  - De ce message, il récupère la localisation du client pour lui transférer cette notification.
3. Invocation d'un service du micro-monde et gestion du retour :
  - Le pont reçoit une demande d'invocation concernant un service de son micro-monde,
  - Du message reçu, il récupère les données nécessaires pour invoquer le service demandé : le fournisseur, le service, l'action et les paramètres d'entrée,
  - Il invoque le service par un appel direct,
  - Si un retour est demandé, il récupère ce que le service a renvoyé et envoie un message NOTIFYRET au routage.
4. Souscription à un type de service par un client et notification : Concernant le pont du micro-monde du client, il ne fait que transférer la demande de souscription au routage et la notification au client.
5. Souscription à des variables d'états associées à un service :
  - Concernant le pont du micro-monde du client, il ne fait que transférer la demande de souscription au routage et la notification au client.
  - Du côté fournisseur :
    - Le pont récupère la demande de souscription,
    - Il souscrit lui-même aux variables du service demandées,
    - Lorsqu'il reçoit une notification de changement d'état, il la transmet au routage.

5.3.3 Le routage des messages

Le routage est la clé de voûte du middleware. Il permet à n'importe quel client d'appeler n'importe quel service sans connaître la technologie employée. Dans notre système, le client envoie un message au pont de son micro-monde. Ce pont transforme le message pour pouvoir le transférer au sein du middleware. Une fois le message transformé, le service de routage le récupère et l'envoie sur le pont correspondant au micro-monde du service invoqué, en s'appuyant sur l'annuaire de service. Le routage se base sur le protocole utilisé par le service, chaque protocole disposant de son propre Topic associé au bridge. Le tableau suivant 5.2 résume les principales règles de routage en fonction des messages en entrée. L'ensemble de ces règles sont disponibles en annexe.

Un client peut souscrire à un service et sera notifié dès que le service deviendra disponible (consécutivement à la demande si le service est déjà présent). Sur le départ d'un service, une notification est également envoyée aux souscripteurs. Un client peut souscrire à un service dont il connaît le fournisseur, ou il peut souscrire à un type de fournisseur et sera donc averti dès qu'un fournisseur du type demandé sera connecté.

5.3.4 Notification sur une variable d'état

Chaque périphérique possède un état à un moment donné. Dans un milieu pervasif, il est intéressant de pouvoir connaître tout changement d'état d'un périphérique que nous utilisons ou que nous souhaitons utiliser. Il est par exemple utile de savoir quand un écran n'est utilisé par personne. Les clients

Message en entrée	Composant appelé	Description
Connexion	Connexion	inscription du client et/ou du fournisseur
DECONNEXION	Connexion, souscription et notification	deconnexion de l'entite et envoie de notifications si besoins
INVOKE et INVOKERET	Annuaire et bridge du service concerne	récupération du bridge par l'annuaire et invocation du service via le bridge
SOUSCRIPTION	souscription	enregistrement d'un client pour l'écoute d'un service
SEARCH	Annuaire	recherche d'un service par un client
ADDFOURNISSEUR	annuaire	inscription d'un fournisseur de service auprès de l'annuaire

TABLE 5.2 – Messages échangés au sein de YMCA

ont donc la possibilité de souscrire à un service pour être notifié de chaque changement d'état de ce service. Pour rappel, un état est caractérisé par un ensemble de variables. Le client reçoit alors le nom de la variable modifiée ainsi que sa nouvelle valeur. Par exemple, si un client change la chaîne d'une télévision, tous les clients s'étant abonnés à cette télévision seront avertis et connaitront la nouvelle chaîne regardée.

5.4 Architecture détaillée

5.4.1 Vue globale

Comme nous l'avons vu dans les parties précédentes, nous avons décidé de nous baser sur OSGi pour construire YMCA. OSGi nous apporte plusieurs choses :

- Une grande souplesse
- Le service Event Admin qui nous a permis de construire notre interigiciel orienté messages
- Ce même service nous facilite la création de connecteurs entre micro-monde

YMCA est composé de différents modules, appelés bundles selon la terminologie OSGi. Chacun de ces bundles a donc un rôle bien défini permettant à l'ensemble de répondre aux besoins cités. Nous allons maintenant détailler l'ensemble des bundles implémentés pour réaliser notre ESB et les services ajoutés. Nous présentons également les différents Event OSGi circulant.

5.4.2 Utilisation d'OSGi : les bundles

Un bundle correspond à l'unité de conditionnement proposé par OSGi. Un bundle correspond à un jar, pour un service ou un ensemble de librairies par exemple, auquel sont ajoutées des méta-données. YMCA utilise la plate-forme OSGI. Chaque service interne ainsi que chaque bridge doit donc être conditionné sous la forme d'un bundle standard OSGi. Les bundles correspondent donc à la description faite dans le chapitre 4.

La plate-forme d'interaction YMCA présentée ici répond donc à nos besoins en communication. Les différents éléments offrent des fonctionnalités de haut niveau qui, reliées ensemble, permettent l'implémentation d'un EVI :

- *la gestion des services* : permet l'ajout, la suppression et la recherche de services
- *les bridges* : permettent l'intégration et l'interfaçage d'éléments, périphériques physiques ou applications logiciels, hétérogènes
- *le routage* : permet de cacher l'infrastructure de communication en traitant le transfert des messages
- *les variables d'état* : permettent de connaitre l'état du système via l'état des différents éléments connectés

La **figure 5.4** représente l'architecture logique de YMCA avec les différents éléments qui la composent.

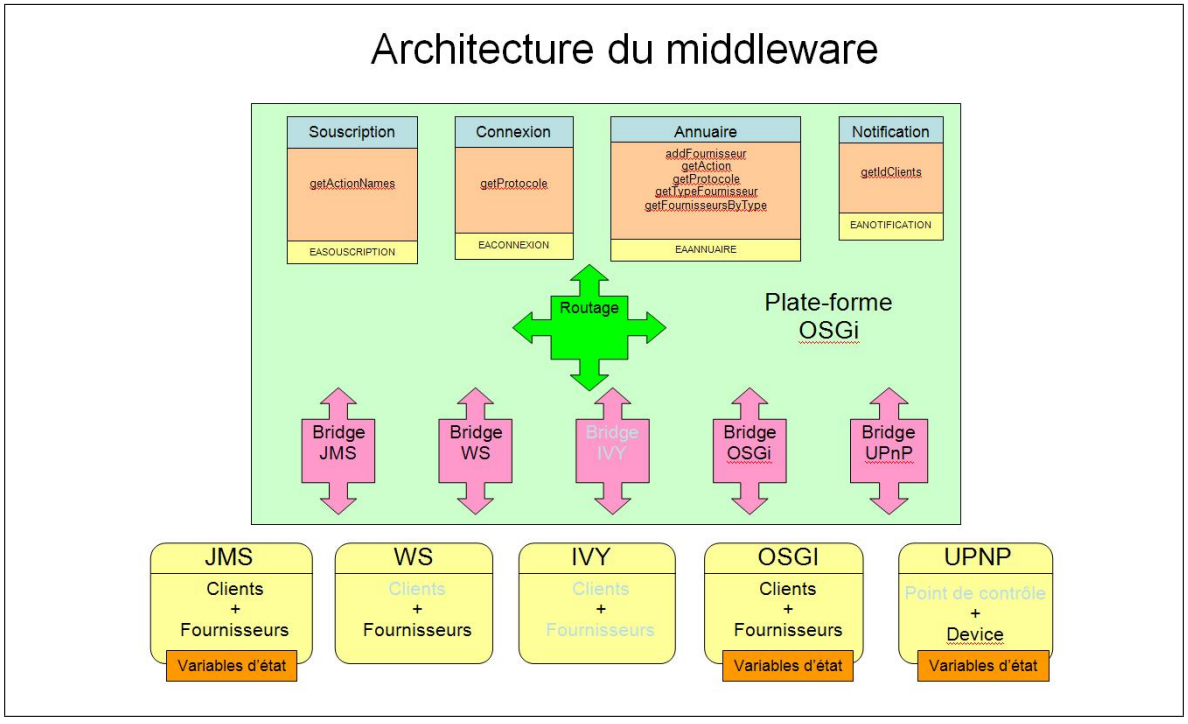

FIGURE 5.4 – Architecture détaillée de la plate forme YMCA

5.5 Les services supplémentaires offerts

Comme évoqué dans les chapitres précédents, les besoins technologiques sont en train d'évoluer. Les progrès techniques récents ont fait apparaître de nouveaux dispositifs et donc de nouvelles façons d'interagir avec la technologie. Les utilisateurs sont en effet de plus en plus mobiles. La multimodalité et la multicanalité sont aussi des facteurs importants dans les nouvelles interactions.

Interfaces tangibles De plus, les dernières tendances montrent une envie de voir se fondre la technologie dans l'environnement quotidien des utilisateurs avec notamment l'utilisation d'interfaces dites tangibles[CD04]. Il s'agit d'utiliser des objets de tous les jours, n'ayant aucun lien avec la technologie afin de les transformer en périphériques d'entrée ou de sortie d'un système informatisé. La force de YMCA est de pouvoir effectivement brancher n'importe quel périphérique moyennant un peu d'électronique.

Réalité augmentée La réalité augmentée [Mac96] est un concept qui couple la perception de la réalité avec un modèle virtuel 2D ou 3D en temps réel. Ce concept se traduit souvent par la superposition d'images ou d'objets 3D dans un flux vidéo qui filme la réalité en temps réel. De nombreux projets traitent de ce sujet, notamment Wikitude [wik]. Notre plate-forme se propose alors d'être un facilitateur pour la création d'environnements propres à la réalité augmentée. Il est en effet facile d'imaginer par exemple dans un EVI un système qui permet d'afficher pour un client le chemin qui mène à un produit, par surimpression de flèches sur le sol par exemple. Au sein des cabines d'essayage augmentées, un système de réalité augmentée peut être installé sous la forme d'un miroir intelligent qui renvoie l'image du client avec des vêtements en surimpression.

TUIO Afin de répondre aux demandes croissantes autour des interfaces tactiles, nous avons choisi d'implémenter un bridge spécialisé dans un protocole dédié aux interactions tactiles : TUIO [KBBC05]. TUIO est un framework permettant de définir un protocole commun et de fournir une API pour les surfaces multitouch tangibles. Le protocole TUIO permet la transmission d'une description abstraite de surfaces interactives, y compris les événements tactiles et les états d'objets tangibles. Pour YMCA, tout dispositif tactile utilisant le protocole TUIO devient un fournisseur de services. Ce service propose une ou plusieurs variables d'état correspondant aux descriptions de la position des curseurs sur l'interface. Ainsi pour réagir aux mouvements d'un utilisateur sur une surface tactile TUIO, un périphérique doit s'être enregistré et avoir demandé la notification sur la surface TUIO. Dès qu'un mouvement est effectué, le périphérique TUIO change ses variables. Tout mouvement tactile sur une surface interactive est donc transmis via YMCA à tous les clients intéressés.

Au cours du projet p-LearNet, nous avons réalisé un prototype d'EVI. Ce prototype, basé sur YMCA, a été déployé en magasin pendant plusieurs mois. Nous avons suivi le scénario décrit en introduction de ce chapitre.

5.6 Déploiement en magasin

La partie qui suit décrit la solution que nous avons déployée et expérimentée dans le cadre du projet p-LearNet. Cette solution a été développée sur la base des principes de notre usine logicielle.

5.6.1 Les étapes de réalisation de la solution

Nous avons effectué une expérimentation au sein du rayon GPS et appareils photos d'un hypermarché du groupe Auchan. Le but de cet expérimentation était la mise en place d'un assistant de vente, un outil aidant les vendeurs dans leurs tâches quotidiennes. Nos essais se sont déroulés en quatre temps :

1. co-conception de l'outil avec les équipes de vendeurs
2. test de 3 mois du premier prototype
3. analyses des retours des vendeurs et co-conception du second prototype
4. test de 3 mois du second prototype

Concernant le premier test, il s'est déroulé au sein d'un même magasin. Cinq vendeurs et leur chef de rayon ont participé à l'expérimentation. Ce test a duré 3 mois et concernait les 5 vendeurs du rayon. Le second test a eu lieu sur deux sites. L'expérimentation s'est en effet étendue à un second magasin avec 4 vendeurs et un chef de rayon. Le prototype de ce second essai a été testé pendant 3 mois également.

5.6.2 Description du PSA

Dans les deux essais, la solution déployée était un prototype de notre Personal Selling Assistant. Au sein de son rayon, le vendeur peut avoir accès à un appareil mobile pour l'aider dans ses différentes tâches quotidiennes. L'étape de co-conception nous a permis de sortir les grandes fonctionnalités auxquelles doit répondre le PSA. Ces fonctionnalités sont :

- l'accès à un ensemble de ressources de formations. Un système d'e-learning classique est proposé par l'I.F.E [24]. Les modules accessibles via un simple navigateur web se présentent sous la forme de didacticiels Flash.
- l'accès à une foire aux questions
- la recherche de fiche-produit dans le SI
- la comparaison de produits
- l'envoi d'informations sur un écran LCD présent en rayon

La communication entre le PSA du vendeur, le SI et l'écran doit être réalisée via un réseau sans-fil fourni par une borne WIFI. Nous retrouvons le scénario défini en introduction de ce chapitre. La solution que nous avons proposé repose sur YMCA : notre intergiciel permet de diriger les demandes du vendeur vers les bons services. Il permet ainsi au vendeur d'envoyer des informations filtrées sur l'écran, d'aller chercher dans le SI du magasin des informations sur un produit ou des ressources explicatives sur des produits ou des technologies.

24. Institut de Formation à l'Excellence, institut de formation propre à Auchan

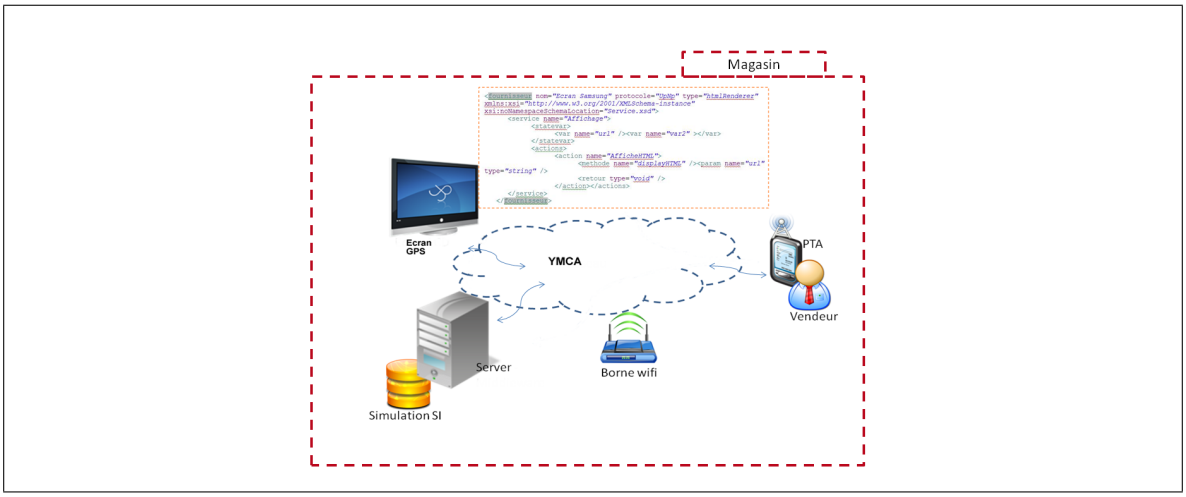

FIGURE 5.5 – Déploiement du premier prototype

Grâce aux services fournis par YMCA, nous avons pu proposer des fonctionnalités supplémentaires, notamment la découverte automatique de l'écran par l'appareil du vendeur. Quand l'écran s'allume, l'application du vendeur le détecte automatiquement et l'application propose alors la fonction de déport d'informations. Et inversement quand l'écran s'éteint, la fonctionnalité disparait.

5.6.3 Architecture déployée

L'architecture déployée pour le prototype est celle décrite dans la **figure 5.5**.

5.6.4 Déroulement des paliers

Ce projet a demandé de nombreuses phases en amont afin de préparer au mieux l'installation du test en magasin. Nous avons eu l'occasion de réaliser deux paliers pour notre prototype. Pour chaque palier, nous avons suivi les mêmes étapes de réalisation décrites précédemment : une phase de co-conception, une phase de tests en laboratoire et une phase de déploiement en magasin.

5.6.4.1 Démarche participative

Dans les projets d'innovation, une des principales raisons de l'échec est le rejet par les utilisateurs. Afin de répondre au mieux aux attentes des vendeurs, nous avons suivi une démarche participative tout au long du projet. Dans un premier temps, nous avons animé des réunions d'informations et de présentations. Puis nous avons étudié leurs différentes activités. Ensuite nous sommes revenus vers eux afin de co-concevoir avec eux les outils que nous leur fourniront. Nous avons étudié avec eux :

- les services proposés par les outils qu'ils jugent utiles pour leur apporter du soutien
- l'ergonomie générale des applications et des périphériques utilisés

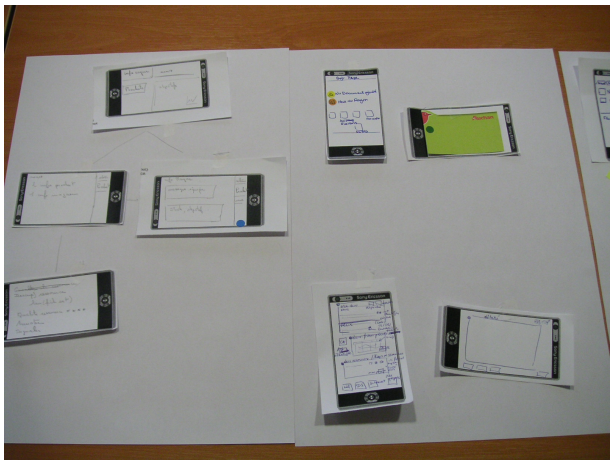

FIGURE 5.6 – Résultat d'un brainstorming autour du PSA

Ces réunions se déroulaient sous forme d'ateliers de brainstorming. La **figure 5.6** représente le résultat d'une de ces séances dédiée à l'IHM attendue par les vendeurs pour une application mobile.

5.6.4.2 Tests en laboratoire

Lors de l'élaboration du prototype, nous avons travaillé en collaboration avec un laboratoire d'ergonomie, EVALAB. Ce laboratoire est spécialisé dans les tests concernant l'*utilisabilité* d'un système [eva].

Les tests d'utilisabilité reposent sur l'observation expérimentale et reproductible des interactions entre cette application et les utilisateurs cibles dans un environnement contrôlé et simulé. Pour cela le laboratoire dispose d'un ensemble d'éléments permettant ces observations. Une salle d'expérimentation est à notre disposition pour recréer un environnement de test et étudier le comportement d'utilisateurs selon des scénarios définis. Cette pièce est équipée d'une caméra dôme au plafond permettant d'observer l'ensemble de la pièce. Cette salle est de plus équipée d'un micro d'ambiance et de 4 cameras. A cette salle, est associée une régie. Par un miroir sans tain, l'ensemble de la salle d'expérimentation est visible de cette régie. Les caméras peuvent aussi être commandées de là. La **figure 5.7** permet d'avoir un aperçu de la console de visualisation des caméras;

Lors de cette séance d'essais, les vendeurs ont émis des craintes sur le projet. Leurs craintes concernaient le côté éthique du projet, car ils se demandaient si le but de notre périphérique n'était pas à terme de remplacer le vendeur. Mis à part ce côté là, le système proposé ne présentait pas de grosses anomalies ergonomiques ni fonctionnelles.

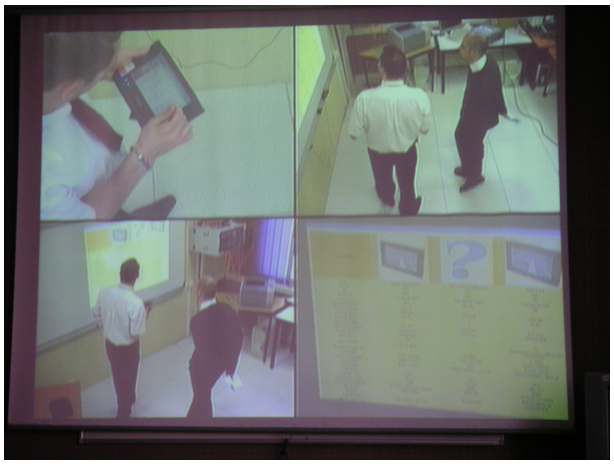

FIGURE 5.7 – Expérimentation en laboratoire

5.6.4.3 Premiers retours

Les premiers retours ont été positifs. Les vendeurs se sont rapidement appropriés l'outil et se sont surtout aperçus que le but n'était pas de les remplacer mais bien de les aider dans leur tâche. Le déport d'écran a été une fonctionnalité bien perçue par l'ensemble des vendeurs, même si certains ont craint un effet *attroupement* qui ne s'est finalement pas manifesté.

Les premiers avis négatifs concernaient le poids et la taille du périphérique, jugés trop importants par la plupart des vendeurs. Ensuite, les vendeur sont revenus sur le côté fonctionnel de l'outil : les modules de formations fournis ne sont pas adaptés à ce genre de périphérique car trop longs ou illisibles. De plus, il manque un accès direct à Internet pour répondre à des besoins clients. Ces premiers retours ont permis également d'affiner les services proposés : les critères de recherches ne sont pas assez exhaustifs, la FAQ ne reflète pas la réalité du terrain, ...

La fonctionnalité d'envoi d'informations vers un écran pour le partage avec les clients a été bien reçue et très utilisée. Mais le contrôle de l'écran n'était pas intuitif. Les vendeurs devaient explicitement relâcher le contrôle sur l'écran et si un vendeur l'oubliait, l'écran devenait inutilisable. De plus, les vendeurs n'avaient pas le contrôle sur les informations produit que l'écran affichait. La configuration était centralisée pour tous les vendeurs.

Certains vendeurs ont aussi proposé un accès libre-service au client sous forme de bornes interactives, le but étant de guider le client dans ses choix puis de faire appel à un vendeur pour les dernières phases de l'acte de vente.

5.6.4.4 Le second palier

Les services proposés par notre système n'ont pas beaucoup changé entre les deux prototypes.

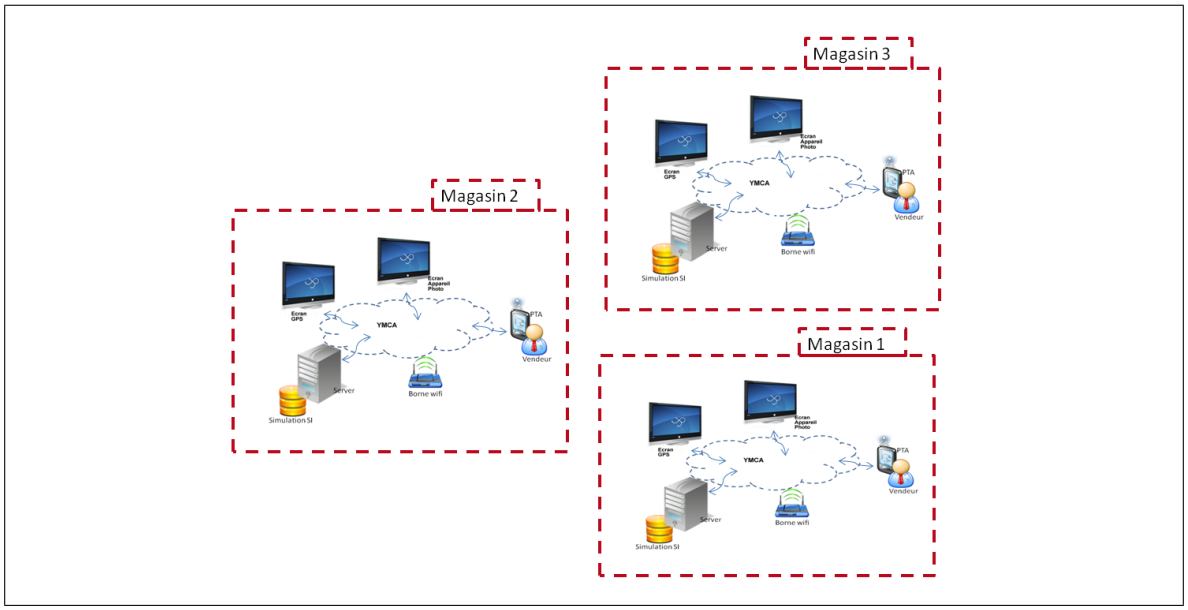

FIGURE 5.8 – Déploiement du second prototype

Les différences entre les deux essais se situent au niveau du périphérique utilisé, de l'IHM fournie et des modules de formation proposés. En plus de fournir un UMPC, certains vendeurs étaient équipés d'un smartphone, le HTC Hero.

D'un point de vue architecture déployée, lors de la première expérimentation, la solution était déployée sur un seul magasin, avec un seul écran proposant ses services. Dans ce second palier, trois magasins ont servi de lieux d'expérimentation. Chaque magasin était équipé deux écrans LCD. Nous avons alors déployé une instance de YMCA dans chaque magasin, chaque instance étant indépendante des autres. Le **schéma 5.8** représente la solution du second palier.

5.6.5 Bilan

À la fin du projet p-LearNet, nous avons pu dresser un bilan grâce à nos prototypes. D'un point de vue utilisateur, les retours ont été très positifs. Les vendeurs se sont rapidement appropriés l'outil, notamment lors de vente avec un client. La possibilité de partager des informations sur un écran est une fonctionnalité très appréciée par les vendeurs mais aussi par les clients.

D'un point de vue technique, la plate-forme YMCA a répondu au cahier des charges. Lors des deux paliers, nous avons pu facilement réaliser le prototype d'EVI. Les écrans répondaient rapidement aux commandes des vendeurs. Le temps de réponse moyen est inférieur à 250ms. Le second palier a permis de montrer la possibilité de brancher plusieurs fournisseurs de services ainsi que des périphériques hétérogènes(UMPC et smartphones) [PBV+12].

D'un point de vue conception, l'ajout de nouveaux périphériques nécessite tout de même l'écriture de morceaux de code : description XML des services, code applicatif pour l'affichage des écrans, écriture des messages permettant d'appeler les services, ...

5.7 Conclusion

Ce chapitre permet de débuter la présentation de notre usine logicielle. En effet, la plate-forme YMCA est le premier élément de notre usine. YMCA permet de répondre aux besoins de l'informatique pervasive grâce notamment à une gestion du contexte (via le système de variables d'état) et à l'hétérogénéité des composants pouvant s'y connecter. Les composants d'EVI peuvent alors interagir entre eux, ou non selon le contexte. Des composants peuvent également être ajoutés dynamiquement et être utilisés sans besoin de redémarrer tout le système. Cette plate-forme a pu être éprouvée pendant 6 mois grâce aux prototypes de PSA réalisés pour le projet p-LearNet. Il reste encore à répondre aux problématiques liées à l'industrialisation de notre solution. Pour les prototypes, nous avons dû coder manuellement tous les éléments nécessaires à l'exécution :

- description XML des services des différents périphériques (e.g. l'écran de déport)
- code Java applicatif
- messages de communication utilisés entre les écrans et les PSAs, pour le déport d'affichage des données
- messages de communication vers le système d'information, pour les fiches produits notamment

Ces éléments sont le genre d'éléments pouvant être générés automatiquement par une usine logicielle dédiée aux EVI. Le chapitre suivant présente l'intégralité de notre usine logicielle, de la modélisation à l'exécution.

Chapitre 6

L'usine logicielle

Le chapitre précédent nous a permis de présenter la première brique de notre usine logicielle. Ce chapitre permet de détailler le fonctionnement complet de notre usine, de la modélisation à l'exécution des éléments modélisés. Dans un premier temps, nous allons définir le principe d'une usine logicielle puis nous détaillerons notre solution, en commençant par la partie modélisation, dans laquelle nous préciserons les différents concepts. Puis nous présenterons l'éditeur graphique permettant d'instancier ces modèles.

6.1 Principes de notre usine logicielle

6.1.1 Définition

[GS04, LW06] nous donnent la définition suivante :

> Une usine logicielle est une ligne de produits logiciels qui permet la configuration des outils extensibles, des processus et du contenu en utilisant un modèle basé sur un schéma pour automatiser le développement et l'entretien des variantes d'un produit et de ses variantes en adaptant l'architecture, l'assemblage et la configuration des composants du cadre de développement.

Pour résumer, il s'agit donc d'un ensemble d'outils facilitant la création et la configuration de logiciels. Cet ensemble se base sur un schéma qui est ensuite implémenté sous forme de template. Ce template est mis à disposition des utilisateurs de l'usine logicielle pour créer leurs propres logiciels. Dans notre contexte, nous voulons fournir un cadre de développement permettant de créer, ou de simplifier la création d'environnements de vente intelligents, comme décrits dans les parties précédentes. Le schéma de notre usine logicielle est décrit dans la **figure 6.1** ci-dessous. Nous pouvons distinguer trois grandes parties :

1. la modélisation : permet de définir les concepts, classes, entités, ...qui vont être manipulés, ainsi que les relations qui existent entre eux

2. le modeleur : correspond à un environnement de programmation graphique, basé sur les modèles, permettant de créer facilement un EVI
3. la plate-forme d'exécution : fournit un ensemble d'outils, de logiciels, de protocoles, . . .permettant d'exécuter les éléments générés par l'éditeur graphique : plate-forme d'invocation de services, annuaire de services, base de données, . . .Notre plate-forme est *YMCA*, présentée au chapitre précédent

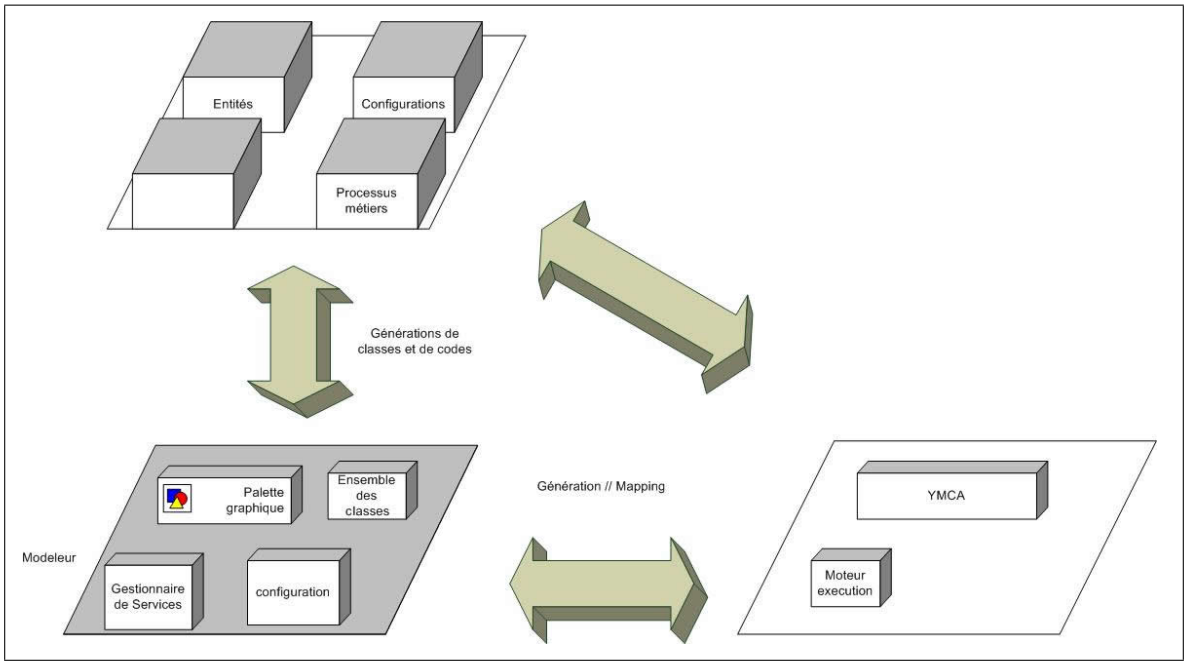

FIGURE 6.1 – Schéma de l'usine logicielle

6.1.2 Une fabrique logicielle pour les besoins de l'informatique pervasive

La fabrique logicielle proposée ici présente la particularité de s'orienter vers l'informatique pervasive, contrairement aux fabriques que l'on peut trouver actuellement sur le marché. Cette orientation vers l'informatique pervasive se traduit par :

- des méta données décrivant une sémantique propre à l'informatique pervasive
- la possibilité d'orchestrer automatiquement et dynamiquement des services
- des interactions dynamiques
- la possibilité de proposer de nouvelles formes d'interactions entre les éléments du système et les utilisateurs (commandes vocales, écrans tactiles, . . .)

Dans nos travaux, le e-retail est un champ d'application de l'informatique pervasive. C'est pour cela que nous appelons notre domaine d'étude le p-retail, pour pervasif retail. Le vocabulaire associé aux modèles définis doit donc être adapté à nos cas d'utilisation et donc au monde du p-retail. Afin de mieux

expliquer notre démarche de conception de l'usine logicielle, nous allons présenter nos premiers travaux. Ces travaux, sous forme de réflexion, ont donné comme résultat des cartes heuristiques exprimant les grands concepts des EVIs.

6.2 Point de départ de la modélisation : des cartes heuristiques

La première étape de notre travail consiste à définir ce que nous entendons par Espace de Vente Intelligent. L'usine logicielle ne peut avoir de sens uniquement si le domaine d'application est correctement modélisé. Le modèle permet en effet de définir ce qui sera concrètement généré. Nous nous appuyons sur l'expérience de l'équipe NOCE dans le domaine des espaces de vente intelligents : vitrine interactive, établi augmenté, cabine augmentée, ...

Les travaux menés par Alain Derycke autour de la cabine d'essayage en réalité augmentée ont servis de point de départ pour une réflexion globale concernant les espaces de vente. À partir de ces travaux, nous avons dégagé deux axes d'études :

1. les actions pouvant être réalisées au sein d'un EVI par des clients ou des vendeurs. La **figure 6.2** représente la carte heuristique des services que nous avons identifiés pour une cabine d'essayage augmentée. Ces actions peuvent être des services rendus par un meuble intelligent comme la cabine (sélectionner/rechercher un produit), ou des actions réalisées autour du meuble (animer la cabine).

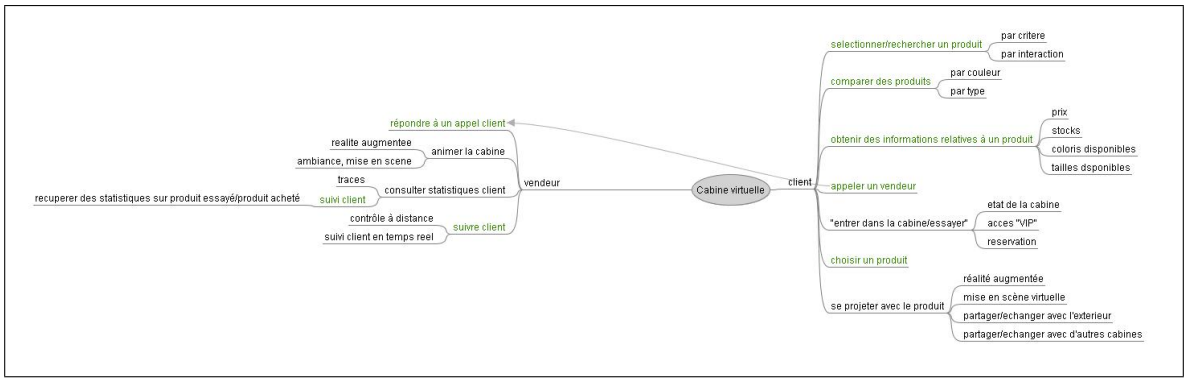

FIGURE 6.2 – Classification des actions autour de la cabine

2. les rôles (objectifs/fonctions) que doit remplir le meuble intelligent par rapport au domaine de la vente. La **figure 6.3** représente la carte heuristique des rôles que nous avons identifiés pour une cabine d'essayage augmentée : faciliter la démarche du client, faciliter l'utilisation par un client, ...

Des travaux similaires ont été réalisés sur d'autres EVI, notamment sur une vitrine augmentée. Nous avons alors dégagé des invariants, points communs pouvant se retrouver dans la définition de plusieurs EVI. Nous avons classé ces invariants en fonction du domaine qu'ils recouvrent. La seconde partie de ce chapitre est dédiée à la présentation de nos travaux autour de la vitrine.

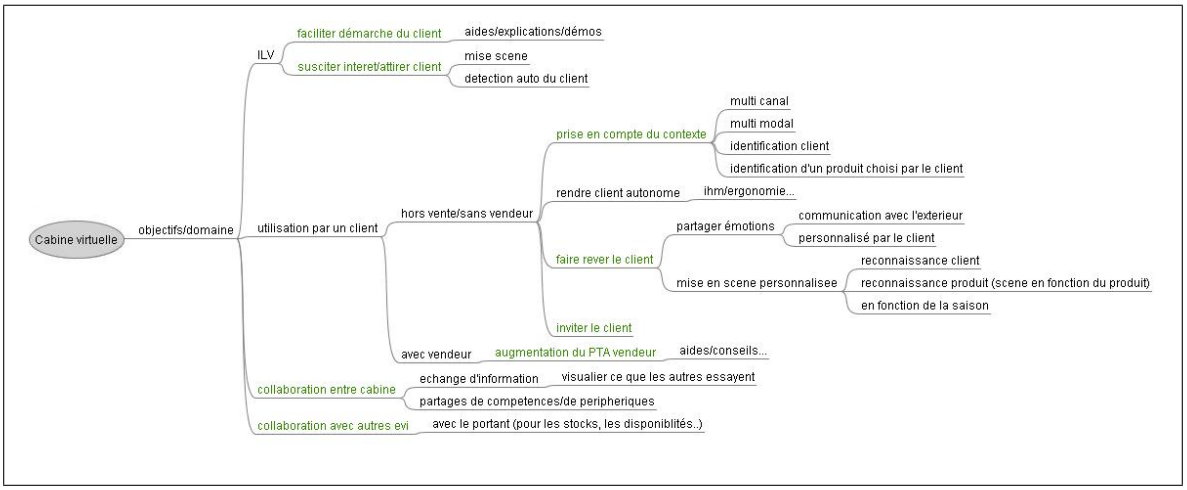

FIGURE 6.3 – Classification des objectifs autour de la cabine

La génération automatique d'EVI que nous visons passe par la modélisation au plus haut niveau des EVI et des concepts du domaine qui les entourent. Afin de généraliser nos réflexions, nous avons dégagé une taxinomie des différents espaces de vente que nous connaissons. Une partie de cette classification est présentée dans la **figure 6.4** ci-dessous.

Nous avons aussi classifié dans la **figure 6.5** les principaux acteurs pouvant intervenir dans ces systèmes.

Notre approche s'intéresse également aux actions pouvant être effectuées par les utilisateurs et aux services pouvant être proposés par le magasin et les EVI s'y trouvant. Nous avons classé ces actions et services de manière chronologique en découpant le processus de vente en trois étapes :

- l'avant vente : l'avant vente concerne toutes les actions qui peuvent entrainer une vente, comme le montre la **figure 6.6** (recherche d'informations, attirer le client ...).
- l'acte de vente : l'acte de vente va du premier contact avec un vendeur (conseils) à la sortie du client avec son produit. Nous avons définis des actions correspondant à cette phase dans la **figure 6.7**.
- l'après vente : l'après vente commence dès la sortie du client du magasin et peut concerner les services d'installations à domicile, les réclamations .... La **figure 6.8** reprend les actions liées à l'après-vente.

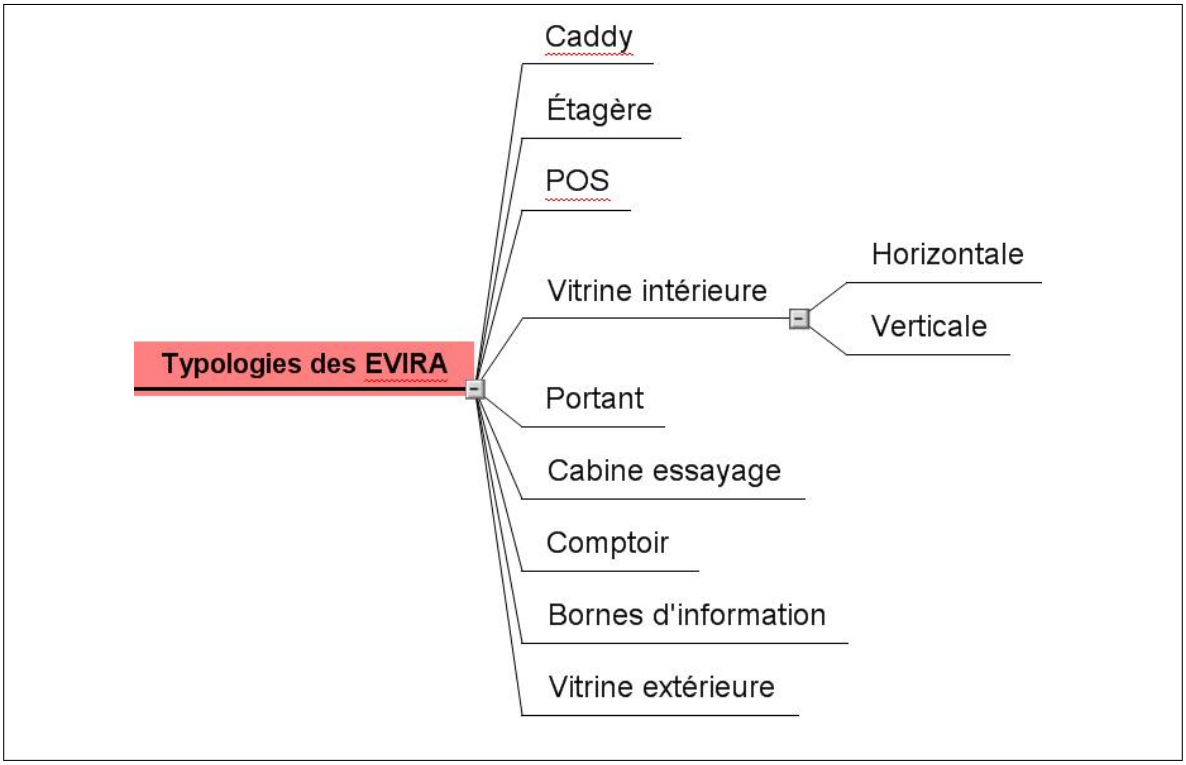

FIGURE 6.4 – Taxinomie des EVI

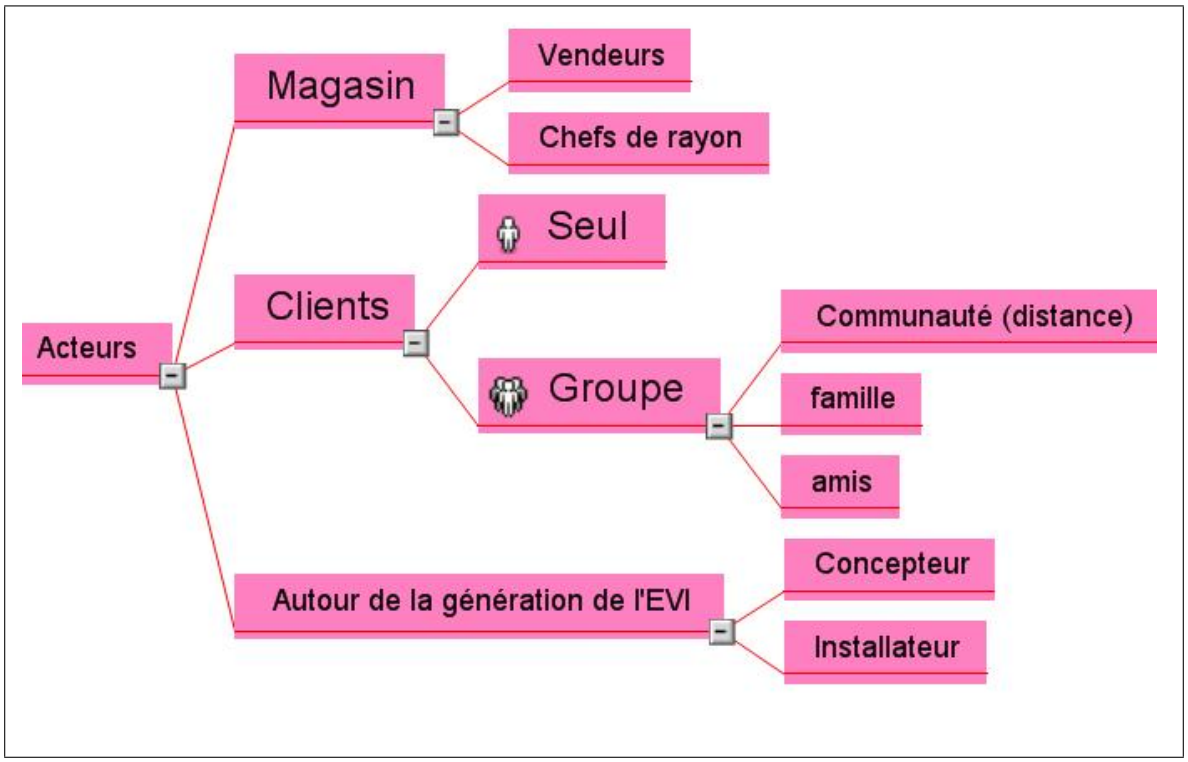

FIGURE 6.5 – Principaux acteurs des EVI

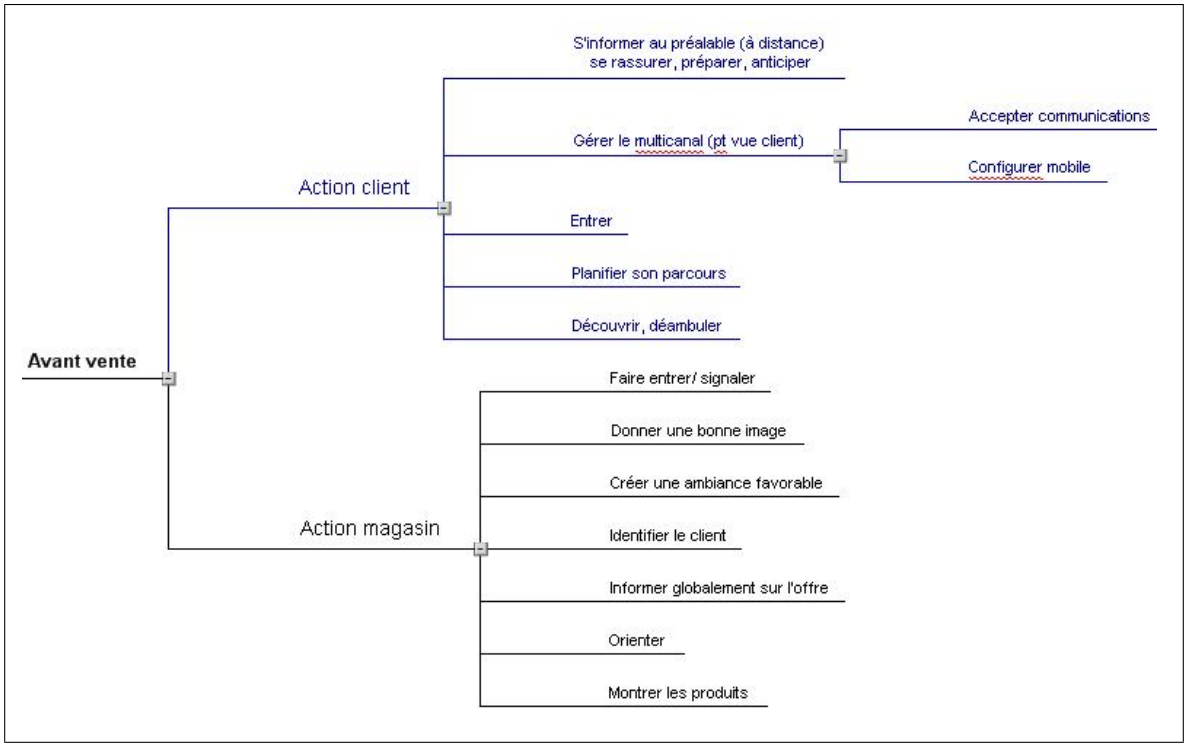

FIGURE 6.6 – Processus d'avant vente

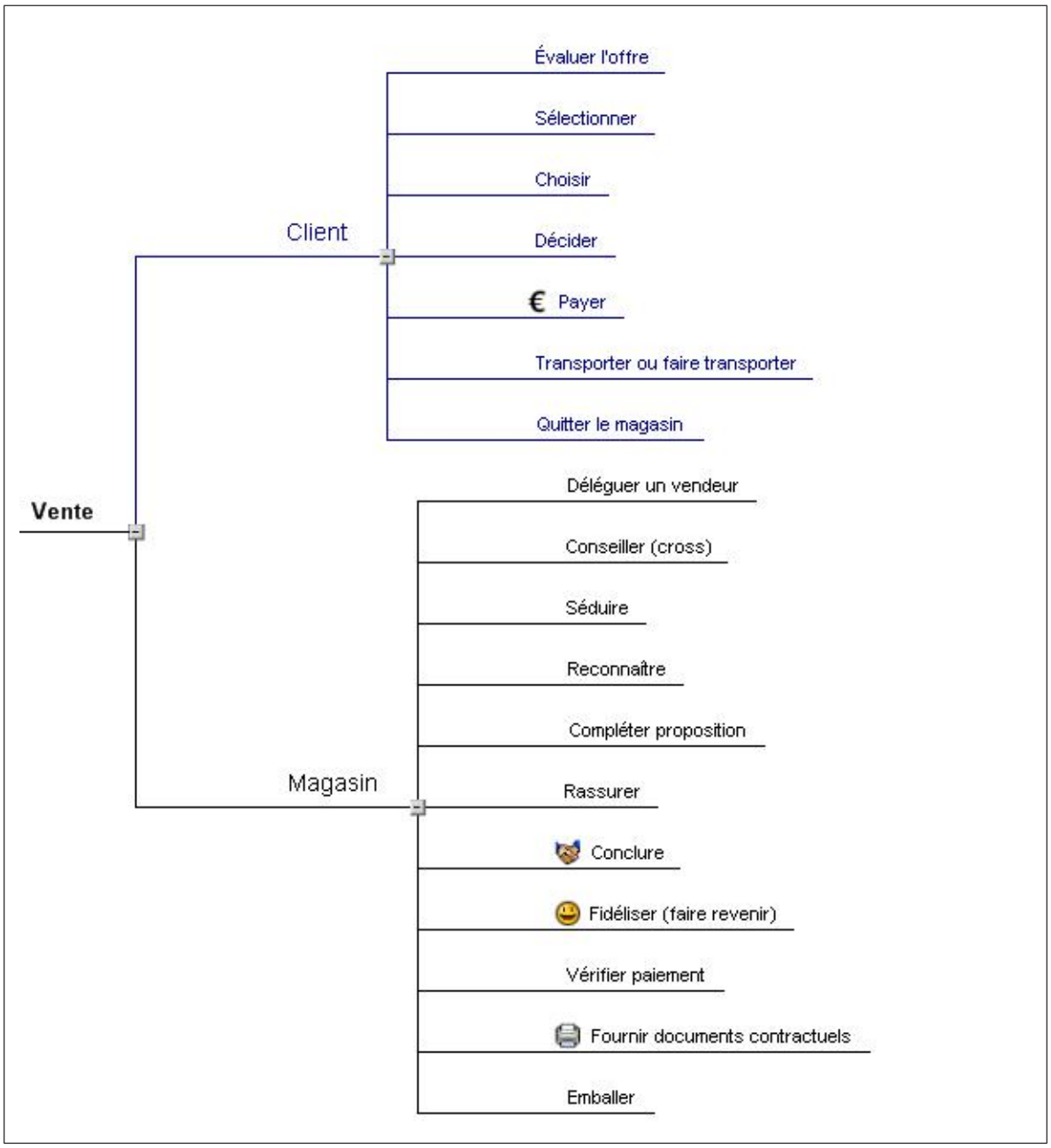

FIGURE 6.7 – Acte de vente

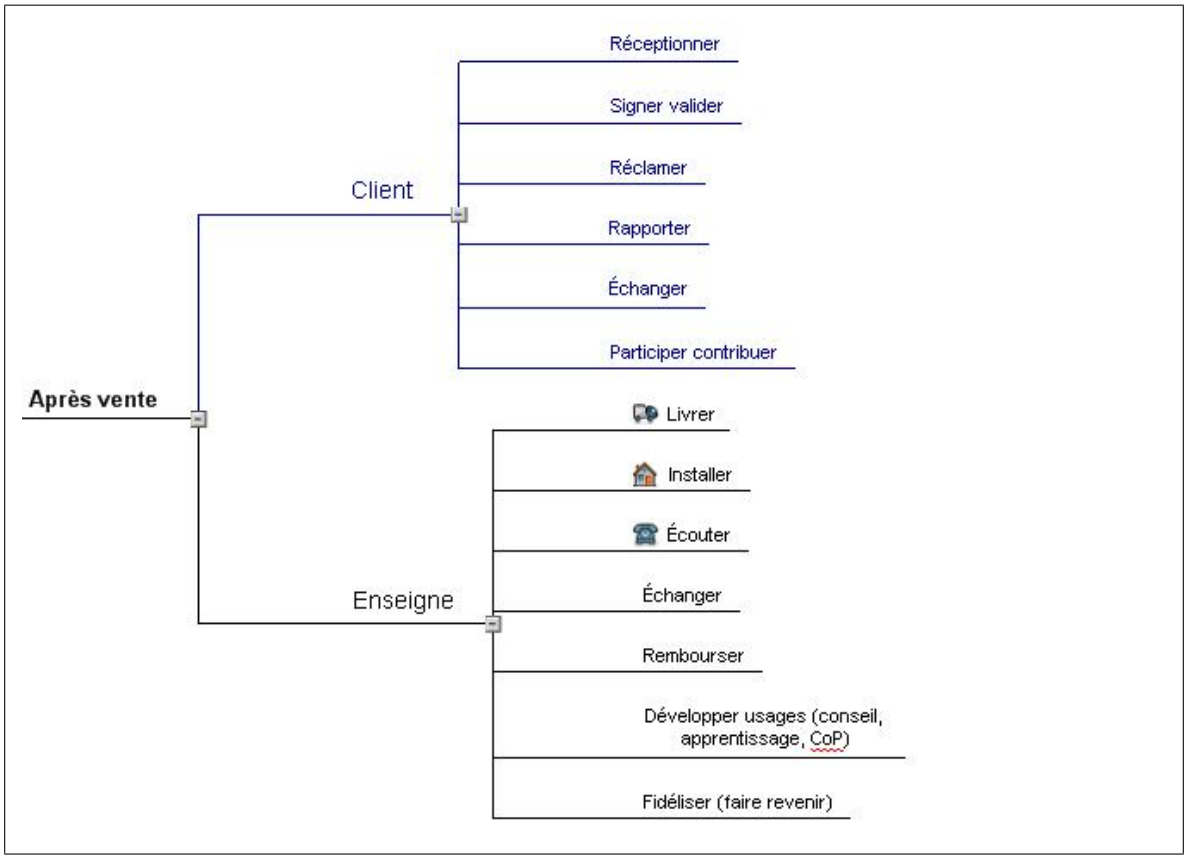

FIGURE 6.8 – Après la vente

6.3 Modélisation d'un EVI

6.3.1 La démarche globale

Si nous reprenons notre contexte global, nous voulons générer un espace de vente intelligent. De manière générale, différents acteurs vont s'atteler à cette tâche à différents niveaux et stades : le concepteur de services/périphériques qui va créer ou insérer des nouveaux périphériques (imprimantes, téléviseurs, lampes, ...) ou des nouveaux services (déport d'écran, comparatif de produits, accès à des sites webs, ...), le chef de rayon qui va donner ses spécifications (services demandés, mobilier choisi, produits visés, objectifs à atteindre, ...), le vendeur qui va personnaliser les différents services et ou interfaces. La liste n'est pas exhaustive car d'autres personnes peuvent entrer dans le processus de création : graphistes, publicitaires, d'autres employés de l'enseigne selon la structure fonctionnelle de l'entreprise (chef de secteur, directeur de magasin, ...).

Pour répondre à ces contraintes, nous avons donc décidé d'utiliser une approche dirigée par les modèles, première étape de l'usine logicielle. Le principe de base de cette approche est de créer différents modèles à partir d'un modèle métier de base, indépendant de toute notion informatique. Ce modèle de base est ensuite transformé en modèle indépendant de toute plate-forme (choix technologiques non spécifiées), enfin ce modèle est transformé à nouveau pour répondre aux contraintes d'une plate forme spécifique pour l'implémentation concrète du système. Le but d'une telle approche est de séparer les spécifications fonctionnelles des spécifications de l'implémentation afin de faciliter l'interopérabilité entre applications. De ce fait, cette approche se veut neutre par rapport aux langages de programmation, aux constructeurs de périphériques ou autres (technologies employées, infrastructures, ...). Nous avons aussi vu un intérêt dans cette démarche quant à la gestion de la participation de différents acteurs. Si on considère qu'un modèle correspond à un point de vue du système, nous pouvons alors, à l'aide de transformations, passer d'un modèle à l'autre en fonction de la personne concernée.Enfin, la modélisation nous permet de gérer facilement la configuration du système. Les objets modélisés peuvent avoir des attributs dont les valeurs peuvent être fixées lors de l'instanciation du modèle. Dans cet optique nous avons alors créé le modèle métier de base. Le modèle de base sert à poser les différents concepts dont nous avons besoin pour générer un espace de vente. Il nous faut donc répondre à la question : qu'est ce qu'un espace de vente? Il faut alors mettre en avant les différents constituants d'un espace de vente intelligent : ce qui le caractérise, les différents acteurs présents, ... De plus, il nous faut modéliser les interactions entre les dispositifs afin de pouvoir recréer un environnement intelligent. Nous avons évidemment découpé ce modèle générique en plusieurs sous modèles afin de faciliter la lecture et la compréhension. Avant de présenter ce modèle, nous allons tout d'abord décrire le méta-modèle dont nous nous sommes servis. Ce méta-modèle explique les concepts de base qui nous servirons à bâtir notre modèle d'EVI.

6.3.2 le méta-modèle d'un EVI

Le méta-modèle permet de définir les concepts de base (entités et relations) qui vont nous servir à décrire les modèles que nous manipulons. La **figure 6.9** présente notre méta-modèle. Dans notre

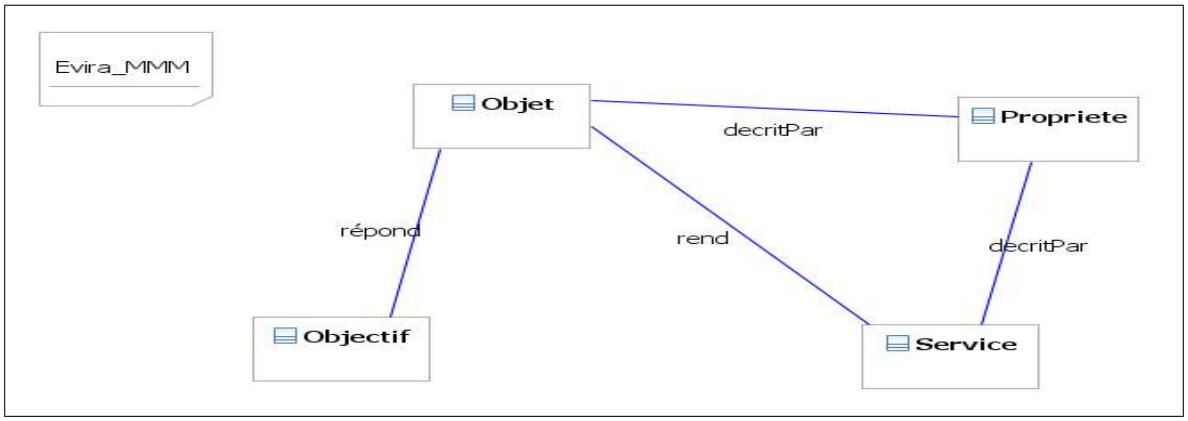

FIGURE 6.9 – Méta-modèle utilisé

système, nous considérons que tout est objet. L'objet est l'élément principal. Un objet est décrit par un ensemble de propriétés, il peut rendre un ou plusieurs services et peut répondre à un ou plusieurs objectifs. Un service est une entité correspondant à une action que l'objet peut réaliser. Nous ne définissons pas de granularité actuellement concernant la définition d'un service. Un objectif correspond à un but à atteindre. Ces buts peuvent être de différentes natures : point final à atteindre, conditions à remplir pendant une période donnée, . . .les services rendus par l'objet peuvent permettre de remplir un ou des objectifs. Dans notre cas d'étude, qui est le e-retail, ces objectifs vont être principalement de deux ordres : financiers et commerciaux. par exemple : attirer le client, fidéliser le client, baisser le coût de perte dû à une mauvaise gestion de stocks, . . .Les propriétés associées à un objet correspondent à des méta-données, c'est à dire un ensemble de données relatives à l'objet et permettant de le caractériser.

6.3.3 Des modèles pour décrire un EVI

À partir du méta-modèle précèdent, nous pouvons maintenant réfléchir à la définition d'un espace de vente. Nos modèles ont été bâtis à partir de nos réflexions présentées dans la partie précédente. Les concepts que nous avons dégagés de nos réflexions peuvent être séparés en deux groupes. Pour décrire le plus finement possible un EVI, nous proposons alors deux modèles :

1. un modèle permettant de décrire la structure organisationnelle d'un EVI et de son environnement, que nous appellerons "modèle structurel"
2. un autre permettant de définir les relations entre éléments, les généralisations . . .que nous appellerons "modèle relationnel"

modèle structurel Le modèle structurel permet de décrire l'ensemble des éléments et les relations d'héritages qui composent un EVI. Nous avons découpé notre modèle en différents sous-modèles selon les concepts utilisés. Nous vous présentons dans les figures suivantes les modèles décrivant les produits présents en rayon, **figure 6.10**, les meubles pouvant être utilisés, **figure 6.11**, le matériel disponible pour enrichir un EVI, **figure 6.12**, et la typologie des services manipulés par notre usine logicielle,

figure 6.13. Cette liste n'est pas exhaustive car il existe aussi le modèle décrivant les acteurs, les objectifs, ...Les figures ci-dessous ne sont en réalité que des parties du modèle. Par exemple pour les produits, il existe encore d'autres produits non représentés ici.

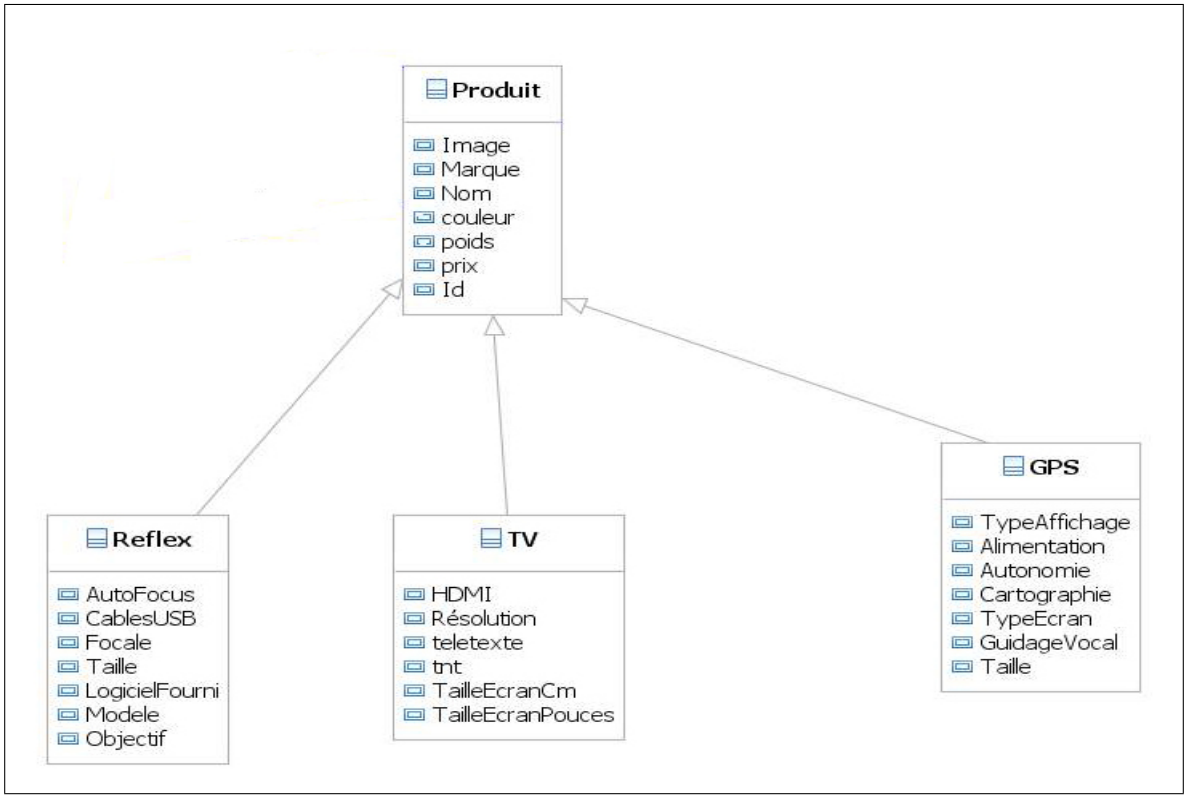

FIGURE 6.10 – Modèle structurel des produits

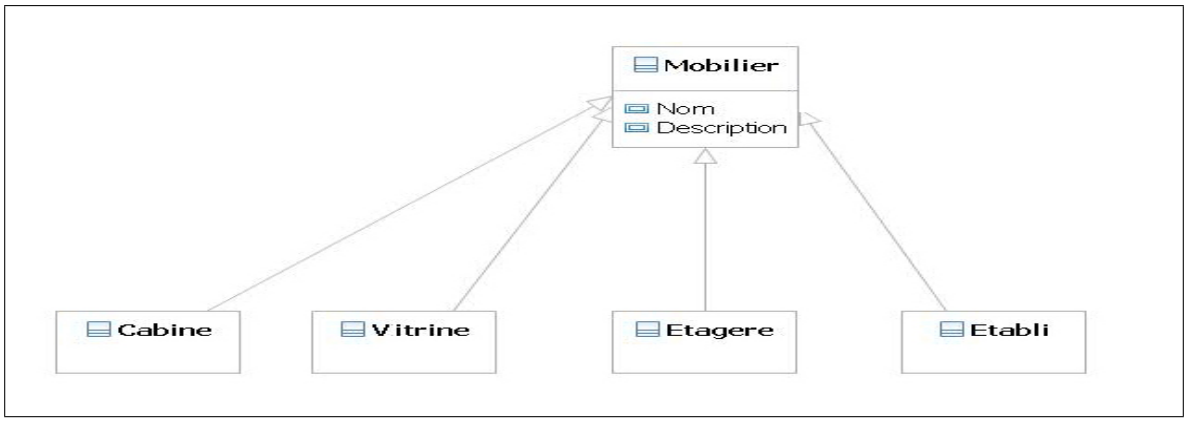

FIGURE 6.11 – Modèle structurel des meubles

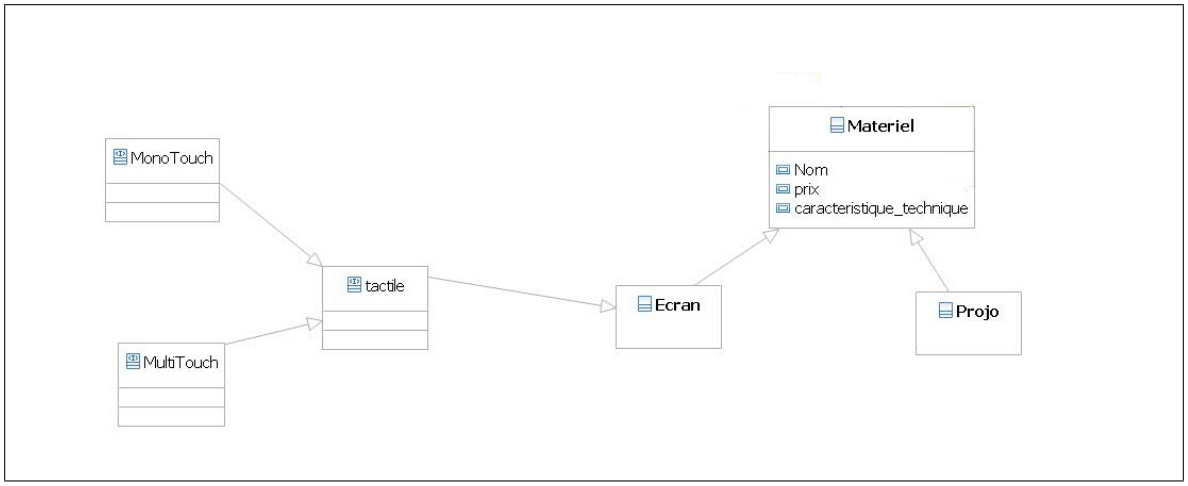

FIGURE 6.12 – Modèle structurel du matériel

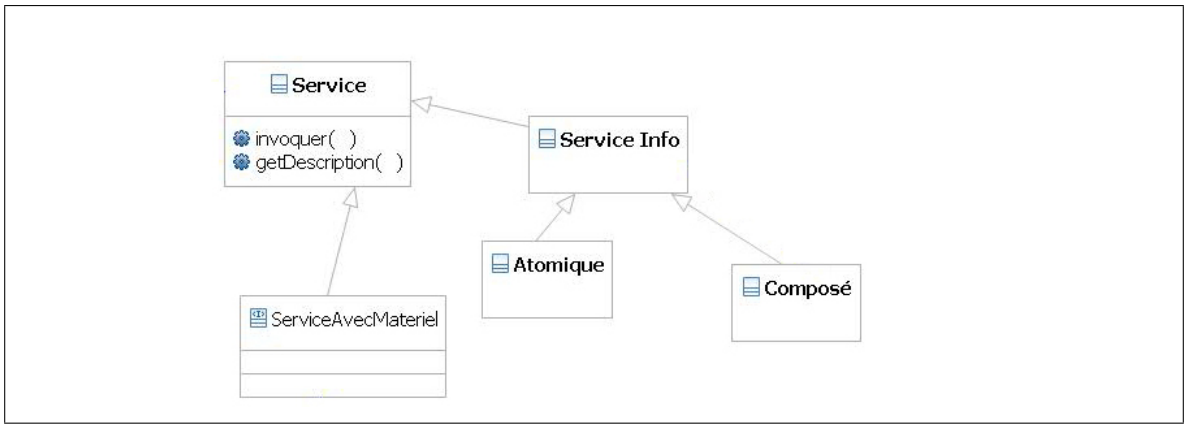

FIGURE 6.13 – Modèle structurel des services

modèle relationnel Le modèle relationnel permet de représenter toutes les relations qui peuvent exister entre les différents concepts d'un modèle. La **figure 6.14** présente les relations des principaux éléments d'un EVI.

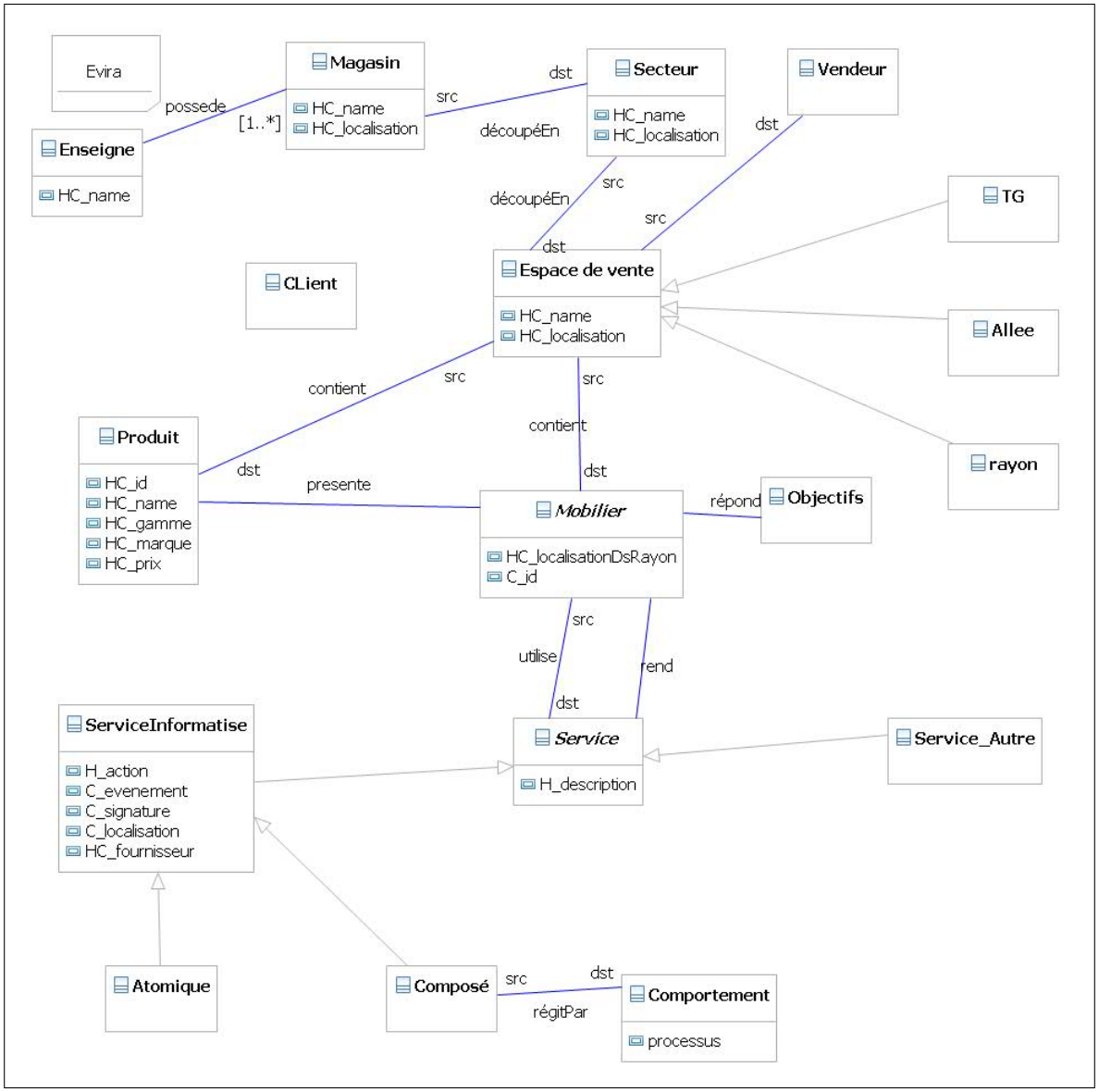

FIGURE 6.14 – Vue globale d'un EVI

Notre modèle relationnel permet de définir un cadre pour nos EVIs : une enseigne possède un ou plusieurs magasins. Un magasin est découpé en secteurs regroupant un ou plusieurs Espaces de vente. Un espace de vente contient un ou plusieurs Mobiliers qui vont présenter les Produits contenus dans l'espace de vente relatif au mobilier. Dans notre modèle, nous avons voulu représenter le fait qu'un Mobilier(concept représentant les meubles) répond à un ou plusieurs Objectifs. Un mobilier

Type	Description
id	un identifiant unique
name	nom d'une personne
productLocation	localisation d'un produit dans un rayon
stock	stock restant d'un produit
price	prix d'un produit
description	fiche technique d'un produit
imageUrl	adresse web pointant vers une image (d'un produit par exemple)
category	catégorie d'un produit
city	nom d'une ville
adress	adresse d'un client
url	adresse internet
accessPointUrl	adresse web d'un service utilisé

TABLE 6.1 – Exemples de types de données manipulées

peut également rendre et utiliser un ou plusieurs services. Ce mobilier pouvant rendre des services informatisés correspond à notre définition de meuble augmenté, permettant à notre espace de vente *traditionnel* de devenir un EVI.

6.3.4 Types de données

Nous avons défini un ensemble de types de données prédéfinies qui seront manipulées par notre EVI. Notre but étant de générer du code, les types que l'on retrouve en langage objet, comme les int, les String, ...se retrouvent également dans nos modèles. Mais nous avons aussi défini des types peu courants comme par exemple le type *productLocation*, pour la localisation d'un produit dans un rayon par exemple, ou encore le type *description* qui permet de donner les caractéristiques d'un produit. En fait, ces types permettent d'ajouter de la sémantique à nos données en précisant ainsi l'usage et la finalité de la donnée manipulée. Ces types prédéfinis vont permettre ensuite au modeleur de proposer automatiquement des liens logiques entre les services de différents meubles et ce grâce au type des paramètres en sortie d'un service et celui des paramètres en entrée d'un autre. Il est donc important de choisir avec soin le type le plus précis et le plus adapté à la sémantique d'une donnée. Le tableau 6.1 présente une liste non exhaustive des types principaux que nous manipulons.

6.3.5 Les outils utilisés pour la modélisation

L'un des problèmes actuels dans cette démarche est le manque d'outils efficaces, robustes et fiables concernant la manipulation des modèles. Pour la réalisation de nos modèles, nous avons utilisé les

outils associés à Eclipse appelés EMF [25]. Le projet EMF est un framework de modélisation et propose une infrastructure de génération de code [VL05]. EMF permet d'adapter des éléments d'un modèle afin de pouvoir les visualiser, les éditer et les manipuler dans un éditeur graphique. Le projet EMF permet de créer deux types de modèles, d'un côté des méta-modèles et de l'autre des modèles instanciant les concepts définis dans les méta-modèles. Tout modèle EMF est une instance d'un modèle EMF avec pour racine commune le modèle *ecore* fourni par EMF. Cela permet non seulement de créer un méta-modèle représentant les concepts désirés par l'utilisateur mais il permet ensuite à l'utilisateur de créer des modèles issus de ce méta-modèle et de les manipuler avec un ensemble d'outils. Les transformations de modèle sont effectuées à l'aide de ATL [26] (langage de transformation) inspiré du standard de l'OMG QVT [27]. ATL est présent lui aussi sous forme de plug-in pour la plate-forme Eclipse.

6.4 Éditeur graphique

6.4.1 Présentation de l'éditeur

L'éditeur graphique, réalisé en AS3 grâce au framework d'Adobe FLEX[fle] [Jay08], est basé sur les modèles générés précédemment. L'AS3 est un langage orienté objet, tout comme java, orienté pour la création d'IHM dite "riche". De plus, l'environnement de programmation du framework Flex, basé sur Eclipse, offre un cadre de développement permettant des développements rapides.

Nous avons choisi d'utiliser des formats de représentation en XML. Ainsi nous arrivons à générer des classes AS3 à partir des modèles XML créés.

Cet éditeur se veut intuitif pour permettre la création par une personne ne possédant pas forcément de connaissances en informatique. L'écran principal, visible dans la **figure 6.15**, se compose des éléments suivants :

1. le menu supérieur
2. le menu latéral gauche
3. le menu contextuel (présent en transparence)
4. le canvas principal

Dans sa version première, cet éditeur permet deux choses :

- la création d'un EVI
- la définition d'interaction entre composants d'un EVI

6.4.2 Création d'un nouvel environnement

Un utilisateur peut créer un EVI comme défini précédemment. Le canvas principal peut être chargé avec un plan de magasin pour choisir afin de définir la position des composants d'un EVI. Dans l'exemple présenté dans la **figure 6.15**, l'utilisateur peut choisir le rayon dans lequel il veut travailler.

25. eclipse modeling framework
26. atlas transformation language
27. Query View Transformation

Les meubles peuvent être ensuite ajoutés, comme le montre la **figure 6.16**, et configurés. L'utilisateur doit pour cela sélectionner des meubles depuis le menu latéral gauche et les faire glisser sur la zone correspondant au rayon. Les meubles sont soit sélectionnés dans une liste, soit recherchés selon différents critères. L'utilisateur peut aussi choisir de générer son propre meuble. Enfin l'utilisateur peut relier des meubles entre eux et permettre à certains meubles d'utiliser des ressources externes, comme un accès à internet ou un accès au SI de l'entreprise.

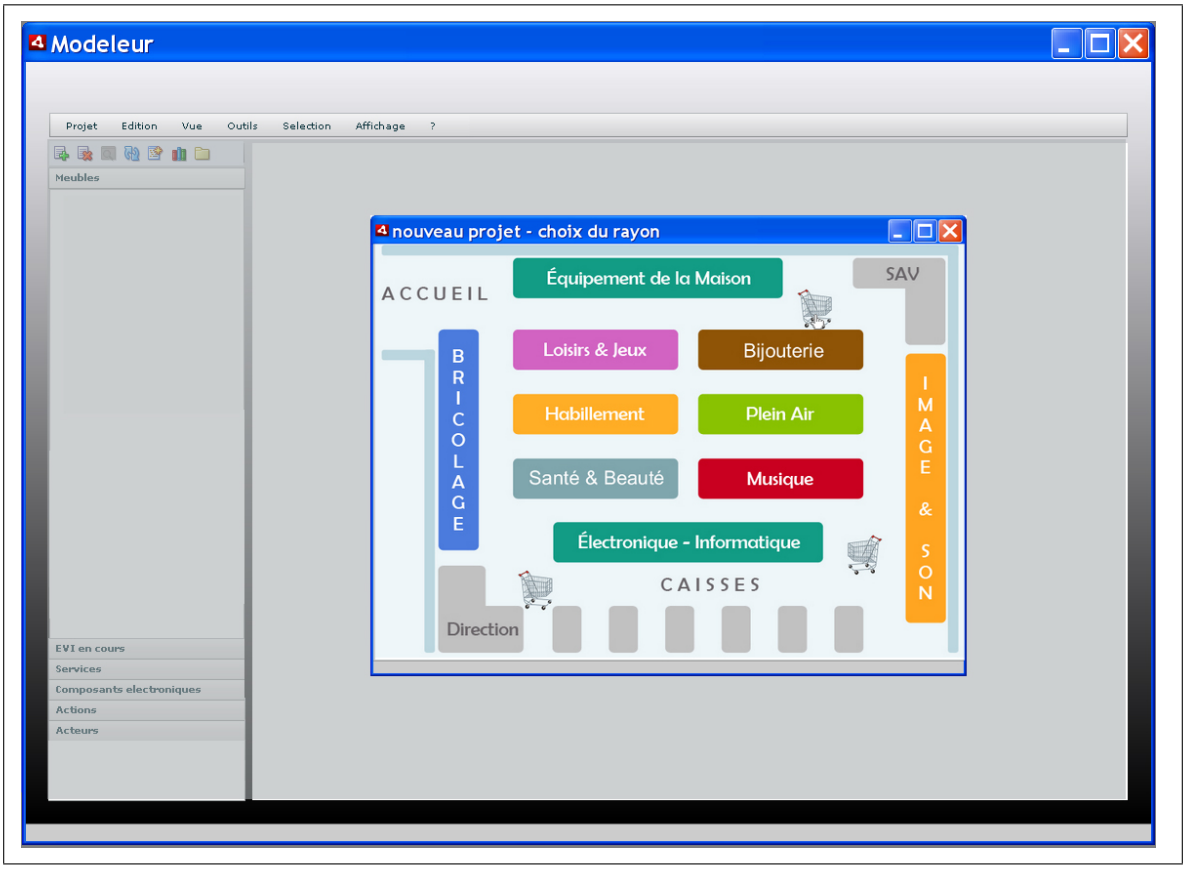

FIGURE 6.15 – Copie écran choix du rayon

6.4.3 Création d'un composant d'EVI

L'utilisateur peut choisir de créer son propre composant d'EVI. La génération de composants consiste à choisir un meuble simple (non interactif) et de lui ajouter des périphériques, des services ou des caractéristiques. Pour choisir ou générer les composants, l'utilisateur dispose d'une matrice à deux entrées : le matériel d'un côté, découpé en catégories, et les services ou caractéristiques, de l'autre, classés selon le domaine. Le tableau 6.2 illustre une partie de ce choix. Ce sont les méta-données associées aux services et aux meubles qui permettent de faire le lien et de proposer des meubles ou accessoires en fonction d'un service ou d'une caractéristique. Par exemple, si l'utilisateur choisit comme

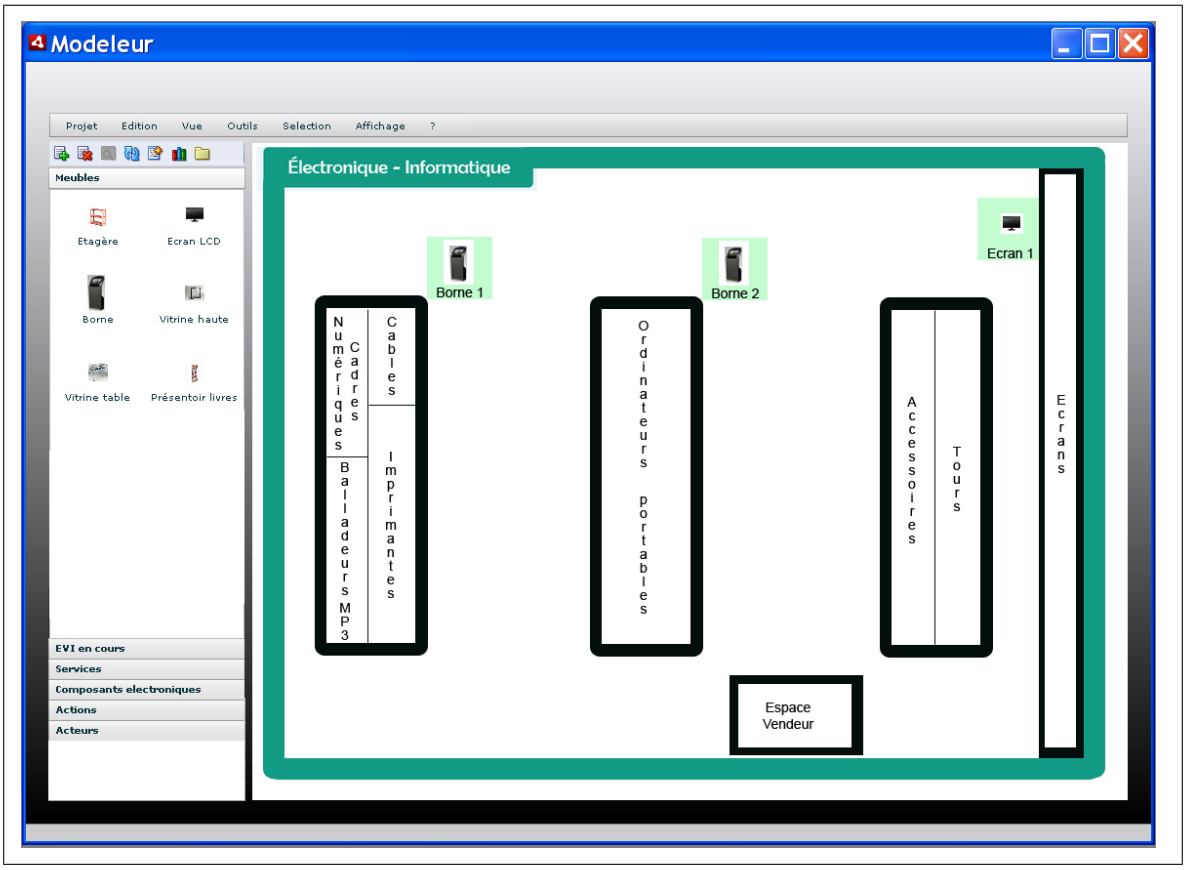

FIGURE 6.16 – Copie écran principal de création d'EVI

caractéristique *interaction tactile*, l'éditeur va lui retourner comme meuble une borne, une vitrine, une étagère ou un écran.

6.4.4 Interaction entre composants d'EVI

La création de liaisons entre les composants d'un EVI permet de définir des interactions entre différents meubles insérés dans l'EVI. Il est également possible de définir des interactions entre les meubles et des services fournis par des systèmes informatisés (SI du magasin, web services externes, ...). Actuellement, seuls les webs services sont configurables dans le logiciel pour être pris en compte. Pour construire l'IHM nous nous sommes basés sur des outils graphiques comme *BPEL Designer*, qui permet de définir des compositions de webs services, et Scratch[SCR]. La liaison entre meubles peut être assimilée à la création de mash-up en liant des services proposés par des meubles. Lorsque l'utilisateur choisit un meuble, il peut, comme dans la copie d'écran 6.18, graphiquement tisser un lien avec un autre meuble. Quand le lien est créé, l'utilisateur peut double-cliquer dessus pour accéder à un écran d'édition de l'interaction. Inspiré par l'outil BPEL Designer, cet écran permet de définir une orchestration de services entre différents meubles ou entre meubles et services externes. Dans l'exemple

	vidéo projecteur	borne	vitrine	étagère	écran
Mise en avant	O		O		
libre accès		O			
Gestion du stock				O	
Afficher de la publicité		O			O
Interaction tactile		O	O	O	O
Localisation d'un produit	O	O	O		

TABLE 6.2 – Matrice de liaison

de la **figure 6.17**, l'utilisateur a configuré le service d'éclairage d'un produit à partir d'une borne interactive. Comme l'illustre la **figure 6.18**, une petite ampoule peut apparaître à côté de certains

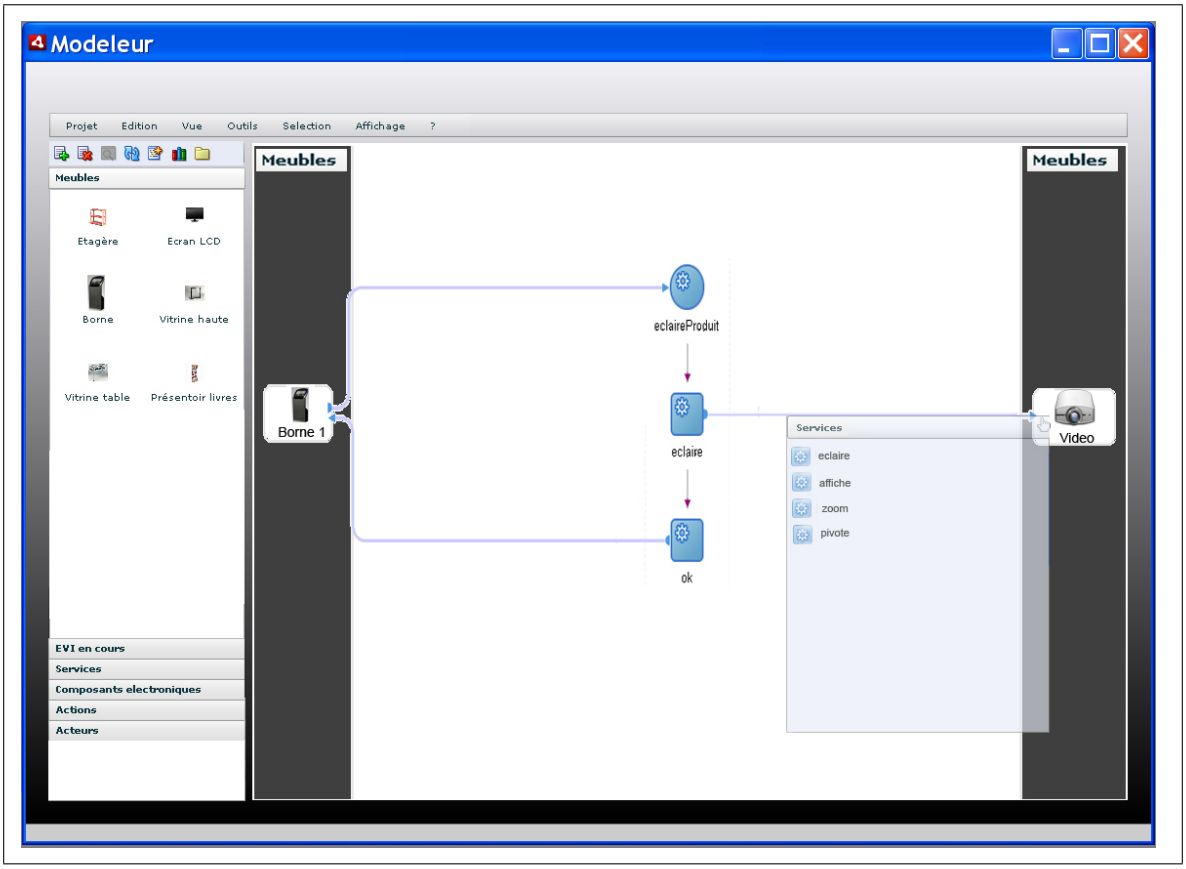

FIGURE 6.17 – Copie écran édition d'interaction

services afin d'indiquer des préférences. Ces préférences correspondent à des liaisons prédéfinies faisant référence à des liaisons types, possibles grâce à l'utilisation de types prédéfinis. Dans l'exemple ci-dessous, l'utilisateur choisit un meuble qui propose de renvoyer la position d'un produit (de type productLocation). Le meuble appelé propose alors un service de mise en avant d'un produit (via un système d'éclairage), ce service est accompagné d'une ampoule pour indiquer à l'utilisateur que la liaison a un sens : le service proposé prend bien en entrée un paramètre de type productLocation. Un message est d'ailleurs associé à l'ampoule pour expliquer que le service peut indiquer la position de l'objet pointé par le service précédent. Ces associations sont définies à l'avance mais peuvent être créées par l'utilisateur et sauvegardées pour être réutilisées. Les services des meubles sélectionnés peuvent être configurés à tout moment en double cliquant sur le meuble souhaité. La liste des services proposés apparaît et il est possible :

- d'activer ou désactiver des services
- de choisir la source de données pour configurer les paramètres d'entrée du service

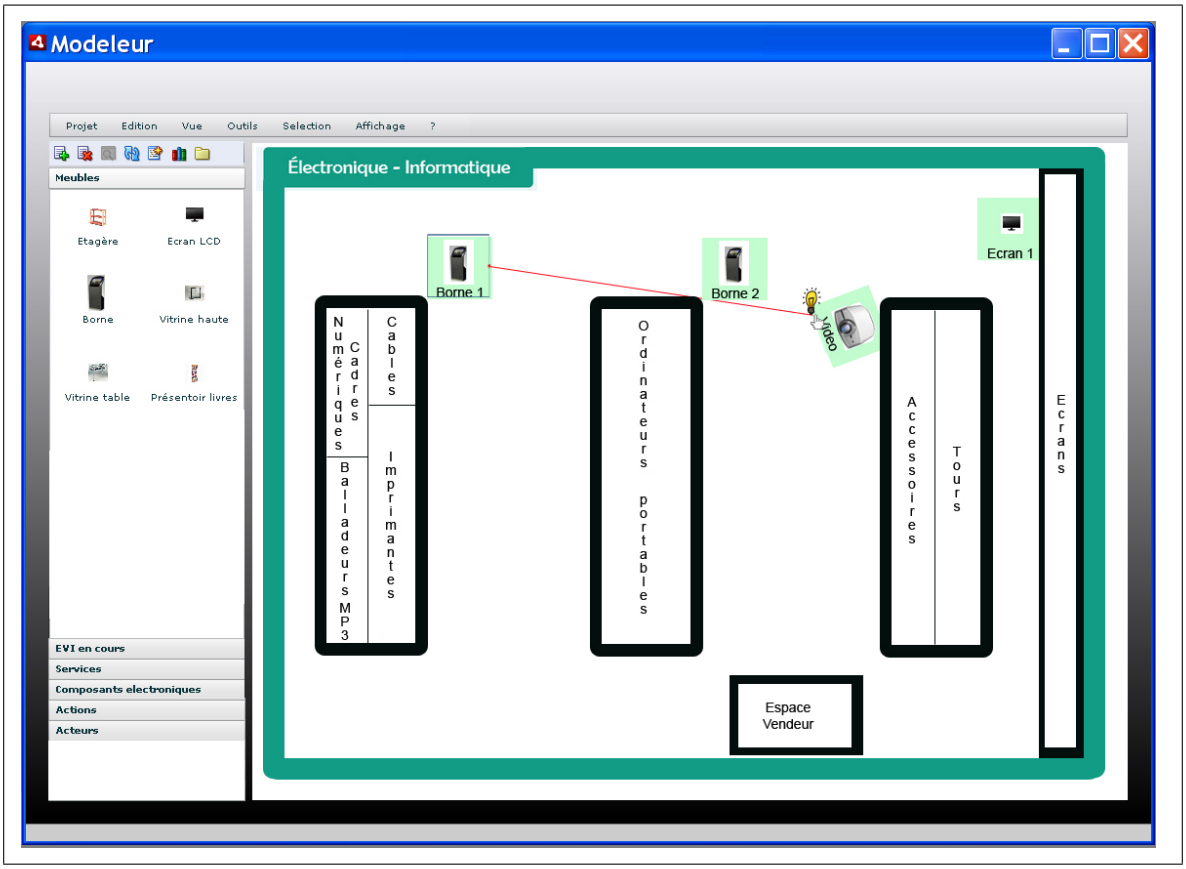

FIGURE 6.18 – Copie écran liaison entre meubles

6.5 De la modélisation à la plate-forme

Les modèles présentés sont utilisés par le modeleur pour permettre la mise en place de l'EVI, d'un point de vue physique et logiciel. Le modeleur génère du code, des morceaux de codes, des recommandations, des schémas ... qui servent à tous les acteurs de la mise en place d'un EVI. Lorsqu'un utilisateur demande la création d'un EVI, les différents éléments modélisés sont alors interprétés par le modeleur pour lui permettre de générer :

- des morceaux de codes correspondants aux différents services proposés par les différents composants de l'EVI
- des schémas électriques, électroniques, ...correspondants à l'installation technique de périphériques spécifiques (capteurs, caméras, haut-parleurs, ...)
- des fichiers de description des différents services et/ou meubles
- des fichiers correspondant aux interactions entre services définis par l'utilisateur

6.5.1 Générations liées aux composants d'EVI

Un composant d'EVI en tant que tel ne génère pas de code. Pour rappel, un composant d'EVI est composé de périphériques et de services. Le code généré va concerné les services qu'il propose ainsi que les interactions avec d'autres composants. Pour rappel, actuellement le modeleur ne génère pas une solution complète. Il sert juste d'assistant pour assembler des services existants. Prenons par exemple un meuble composé d'un service d'allumage d'une LED. Les codes générés pour cet exemple sont alors :

- le code correspondant au service d'allumage et d'extinction offert par la LED via le protocole implémenté par la plate-forme (ici ZIGBEE [zig])
- des explications et des schémas qui permettent d'installer les LEDs et de les relier au code précédent (configuration du code, branchements à effectuer, ...)
- le morceau de code permettant d'appeler le service d'allumage et d'extinction de la LED. Ce code doit être inséré dans le code de l'application qui va gérer le meuble. Cette manipulation est à faire manuellement.

L'ajout de périphérique, comme ici une LED, se traduit par la réutilisation de code pré-existant et correspondant à un comportement standard. Ces parties de codes correspondent à un type de service prédéfini. Si le périphérique utilisé n'est pas connu, ou que son micro-monde n'est pas encore relié à la plate-forme, le modeleur ne génère rien. Nous ne pouvons actuellement ajouter de technologie ou de services à la volée sans devoir développer de composants logiciels spécifiques pour notre plate-forme.

6.5.2 Générations des interactions

Nous partons du principe que les services qui interagissent sont connus et déployés dans la plate-forme *YMCA*. Les interactions entre services correspondent à des orchestrations de service. Il est possible d'obtenir une orchestration de services en développant un service de haut niveau qui pilote les services appelés. Mais cette solution n'est ni souple ni dynamique : il est impossible de redéfinir l'orchestration à la volée, il faut redéfinir le service de pilotage. Pour permettre une orchestration

de service dynamique, *YMCA* propose deux solutions. La première est BPEL [28], langage dédié au départ aux orchestrations de web services. Le modeleur génère pour cela un WSDL qui va être utilisé par le composant spécifique lors de l'exécution. Ce composant va lire le WSDL puis va appeler les différents services selon l'ordre défini dans le fichier WSDL. Les services identifiés dans le WSDL sont des services identifiés dans le registre de la plate-forme d'exécution. La seconde façon d'orchestrer des services consiste à utiliser le framework CAMEL d'apache [apa]. Pour cela la plate-forme propose un composant qui permet de dialoguer avec l'intergiciel de la plate-forme d'exécution. Au sein du framework CAMEL, une orchestration est appelée une *route*. Une fois la route générée, la plate-forme la propose alors comme un service pouvant être invoqué comme n'importe quel autre service.

Ainsi pour générer une interaction, le modeleur va générer soit un fichier BPEL, soit une route CAMEL correspondant aux liaisons définies graphiquement. Le choix du fichier généré est un paramètre à choisir dans l'application.

6.5.3 Recommandations

Une des principales caractéristiques de notre plate-forme est également l'apport d'aide. Cette aide peut se traduire de différentes façons : recommandation électriques, électroniques, mode d'emploi de périphériques utilisés, best practices...Cette aide est fournie grâce à l'ajout de méta-données sur ces documents. Ces méta-données sont relatives aux services et/ou matériels que l'utilisateur de l'usine logicielle a sélectionné.

6.6 Réalisation

Dans le cadre de nos réflexions autour des nouveaux espaces de vente intelligents, nous nous sommes penchés sur l'amélioration de l'interaction entre les objets présentés à la vente et les clients. Un des concepts qui est revenu régulièrement lors de nos séances de mind-mapping est celui de vitrine. Une vitrine est le point de départ de nombreuses ventes. Elle est censée mettre en avant les objets qui y sont présentés pour attirer le client et l'intéresser. Nous nous sommes alors posé la question de l'augmentation d'une vitrine.

6.6.1 La vitrine augmentée

La vitrine doit garder sa fonction de base, qui est de montrer de façon attirante des objets tout en les protégeant. Actuellement, la plupart des vitrines ne proposent aucune interaction. Les objets sont présentés, avec souvent des mises en scène, de manières statiques. Nous proposons alors d'améliorer le concept de vitrine. Le concept de vitrine augmentée est profitable aux clients mais aussi aux vendeurs.

point de vue des clients Cette vitrine doit permettre d'établir un contact d'un nouveau genre avec le client. Elle doit permettre de mettre en valeur les objets qu'elle contient et susciter l'intérêt du client, et se séparer de son rôle premier qui est la protection des articles. Pour cela, elle doit attirer

28. business process expression langage

l'attention du client par son aspect extérieur et susciter sa curiosité grâce aux nouvelles interactions qu'elle propose. Afin d'attirer le client à utiliser la vitrine comme support à l'achat, nous souhaitons rendre les objets manipulables, même à travers la vitre. Une vitre tactile peut être utilisée pour cela. La manipulation des objets peut être restreinte au minimum, par exemple, sélectionner un objet et le faire tourner. La sélection d'un produit peut alors lancer juste une mise en valeur simple de l'objet (surbrillance par un jeu de lumière,) soit démarrer une animation autour de l'objet choisi (vidéo publicitaire par exemple).

La vitrine a par ailleurs pour but d'apporter du rêve au client, en créant une ambiance particulière autour des objets qui sont vendus. Le client doit garder en tête une image positive des produits qu'il a regardés. Enfin, la vitrine interactive est conçue pour faciliter la démarche du client et rendre les informations plus rapidement accessibles en supprimant l'intermédiaire habituel qu'est le vendeur. Nous voulons alors pouvoir donner de l'information sur les objets exposés au client. Ces informations sont affichées sur un écran intégré à la vitrine. Les données affichées peuvent être purement informatives, description de l'objet, prix, ... Elles peuvent aussi être de nature commerciale, orientée cross-selling en donnant des références d'accessoires utiles à l'objet exposé par exemple, ou orientée up-selling, en indiquant les produits de gamme supérieure. La vitrine propose aussi la possibilité de comparer des produits exposés et d'afficher un tableau sur l'écran. Un lien peut aussi être établi avec un site d'e-commerce, pour plus d'informations ou pour passer commande par exemple.

point de vue des vendeurs Ce concept de vitrine augmentée doit aussi profiter aux vendeurs. En rendant le client un peu plus autonome, les vendeurs sont moins sollicités pour des questions n'entrainant pas de vente. Couplé au PSA, la vitrine peut également aider lors de l'argumentaire de vente en suivant les actions réalisées par le client (quel modèle semble l'intéresser, ...). Ainsi, lors de la prise de contact « physique » avec le client, son discours sera directement ciblé sur les attentes de ce dernier, et non pas orienté vers la promotion d'un article en particulier. Afin d'appuyer l'échange d'informations entre le vendeur et le client, nous souhaitons intégrer un écran piloté par le vendeur sur lequel peuvent être affichées des informations. De plus, la vitrine peut permettre à un client d'alerter un vendeur si il en a besoin; Le vendeur est alors averti sur son PSA qu'un client l'attend. D'un point de vue installation, la tâche de disposition des produits est simplifiée grâce au modèle de disposition des articles fournis par l'usine logicielle. Ce modèle sert de template que les vendeurs n'auront qu'à suivre pour confectionner la vitrine. Enfin, les chefs de rayon peuvent aussi s'appuyer sur la vitrine pour mieux connaître leurs produits. L'informatisation de la vitrine permet de journaliser chacune des actions réalisées dessus, ce qui rend possible la production de statistiques détaillant son utilisation et les informations consultées par les client. Les chefs de rayon peuvent donc avoir un meilleur retour sur le taux de produits attirants et réellement achetés.

La figure 6.19 présente une maquette de notre vitrine augmentée.

De l'expression des besoins que nous venons de définir, nous pouvons alors classifier les fonctions de la vitrine en deux catégories : les fonctions commerciales et les fonctions techniques. Nous allons ensuite traduire ces fonctions techniques en services. Ces services sont ensuite utilisés lors la création

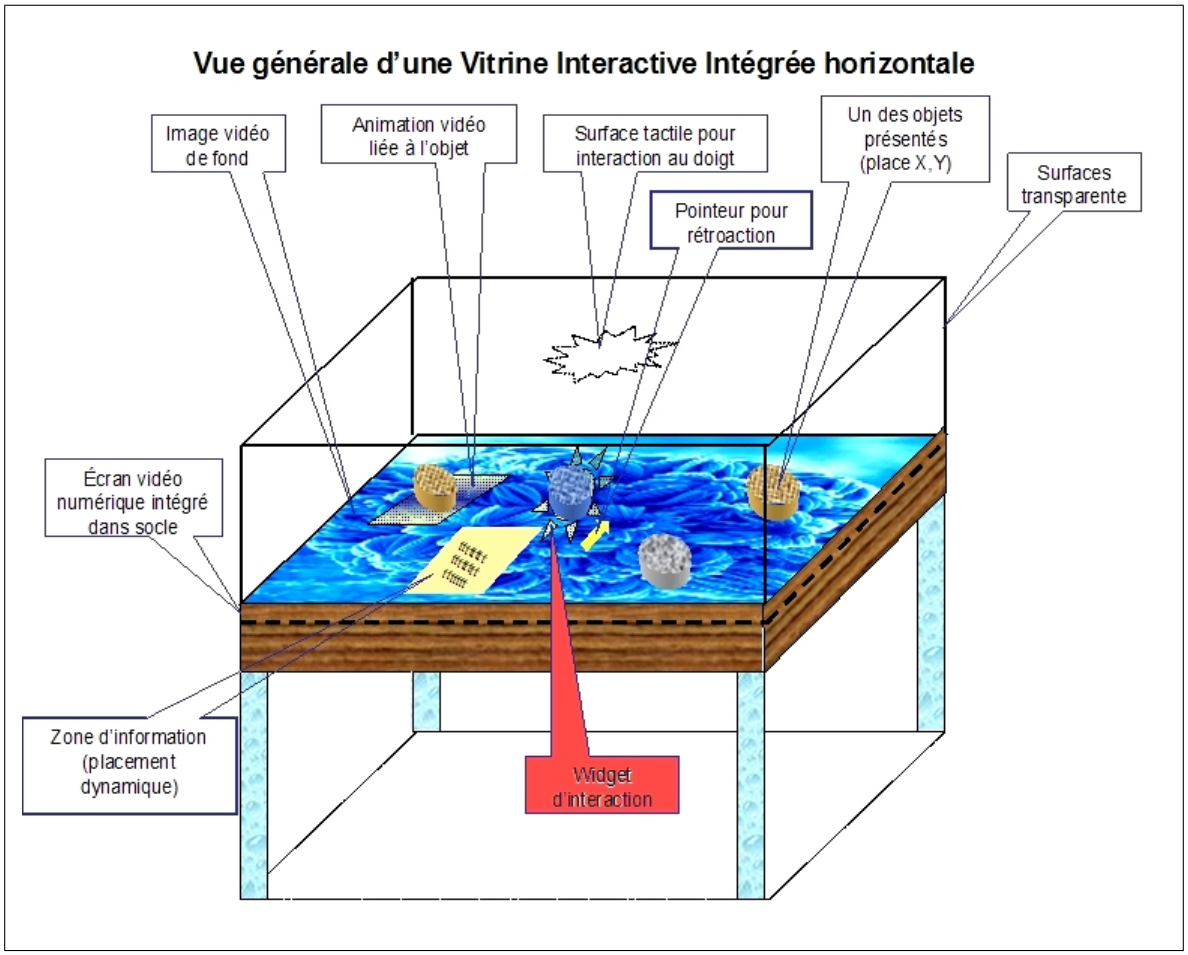

FIGURE 6.19 – maquette de la vitrine augmentée

de l'EVI via le logiciel d'édition. Une fois l'EVI de la vitrine créé via l'éditeur, l'outil effectue une génération des éléments nécessaires à l'implémentation d'une vitrine.

6.6.2 Modélisation de la vitrine

La première étape consiste donc à modéliser notre EVI afin de bien cerner les besoins.

Les fonctions commerciales Les fonctions commerciales dépendent des acteurs qui entrent en jeu dans l'EVI représenté par la vitrine. Les acteurs définis ici sont le vendeur, le client, l'écran et la vitrine. Le vendeur, la vitrine et l'écran doivent proposer des fonctions au client. Les principales fonctions sont :

- conseiller un client
- expliquer des concepts
- mettre en avant des objets

La vitrine peut aussi proposer des services au vendeur, dont notamment :
- proposer des statistiques d'utilisation
- prévenir lorsqu'un client a besoin de la présence d'un vendeur

Quant à lui, le client a besoin des fonctions suivantes :
- sélectionner un produit
- appeler un vendeur
- comparer des produits
- demander des informations sur un produit

Ces fonctions commerciales vont alors amener à des fonctions techniques permettant leur réalisation.

Les fonctions techniques Les fonctions techniques sont rattachées aux différents éléments physiques qui constituent la vitrine augmentée et son environnement, à savoir la vitrine, l'écran et le PSA. Ces fonctions vont correspondre aux services que va proposer l'élément qui remplit la fonction. Seuls deux éléments vont ici proposer des services : la vitrine et l'écran de déport. Pour la vitrine, les services proposés sont :
- mettre en surbrillance un produit
- afficher les informations relatives à un produit
- afficher le comparatif de plusieurs produits
- demander un déport d'écran
- appeler un vendeur
- afficher des statistiques

L'écran propose un seul service : afficher des informations.

6.6.3 Générations et déploiement

Une fois les modèles définis, nous pouvons donc demander au logiciel de générer les morceaux de code qui nous serviront à l'exécution. Nous avons alors, via l'éditeur graphique, généré un EVi avec les différents éléments. Pour chaque élément, nous avons ajoutés les services correspondant aux fonctions techniques définies précédemment. Nous avons donné des précisions sur les services et le matériel utilisé. Par exemple, nous avons donné la description de l'écran comme étant un écran de type HTMLRenderer, comme décrit dans le **chapitre 5**. Nous avons de plus lié la vitrine et l'écran afin de définir la fonction de déport. Pour notre prototype, l'usine logicielle a alors généré :
- les messages qui seront échangés entre la vitrine et l'écran
- les squelettes des classes pour la vitrine, les moteurs et l'écran
- les fichiers de descriptions XML correspondant aux services proposés par l'écran (déport), la vitrine (selection objet, déplacement objet,...)

Nous devons coder nous-mêmes le reste des éléments, notamment l'interface graphique de la vitrine et les contrôleurs qui vont utiliser les messages générés par l'usine en fonction de l'action demandée sur l'IHM. Une fois le logiciel fini, il reste à réaliser le prototype de vitrine. La surface tactile utilisée est

une surface commercialisée par 3M. Un EEE-pc d'Asus est utilisé comme serveur et intégré au sein de la vitrine.

La vitrine ainsi conçue a été présentée au salon de la VAD, à Lille Grand Palais, comme le montre la **photo 6.20**.

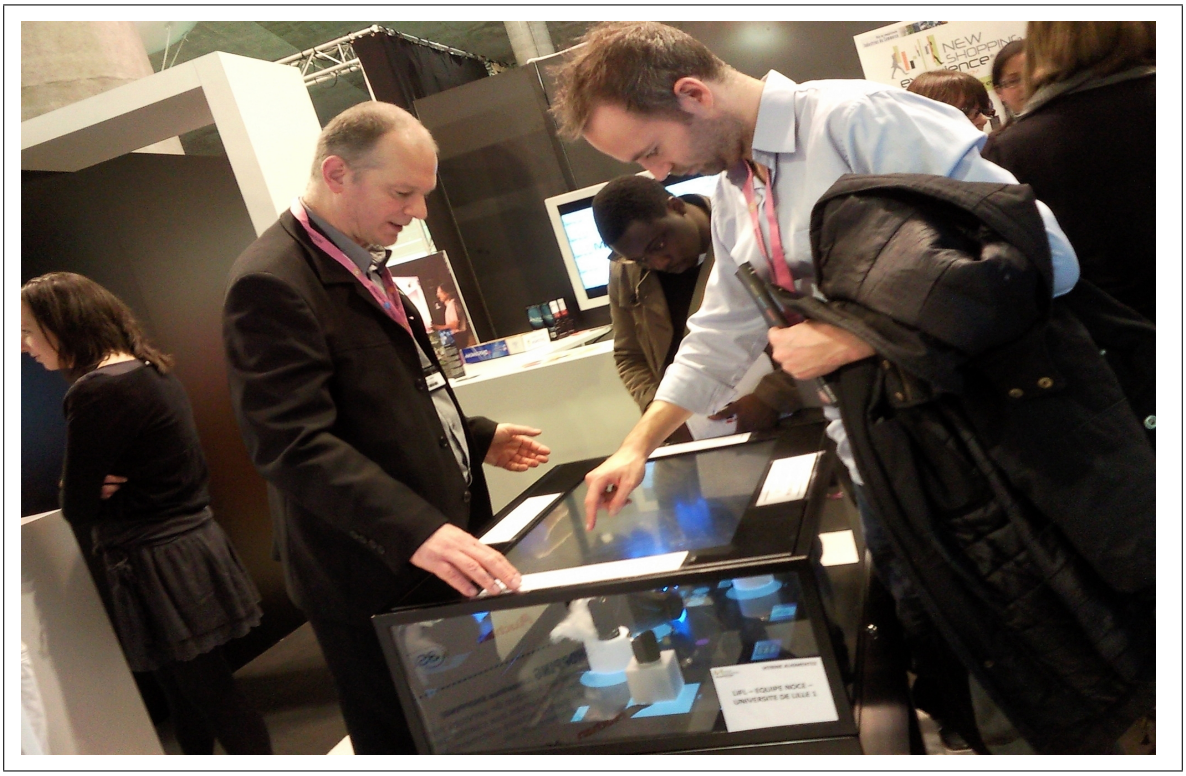

FIGURE 6.20 – prototype de vitrine présenté au salon de la VAD

6.7 Conclusion

Dans ce chapitre, nous avons mis en place les outils permettant de générer tout ou partie du code nécessaire au fonctionnement d'un EVI. La première partie du chapitre est dédiée aux différents modèles que nous avons définis pour représenter notre domaine d'application, les EVIs. La seconde partie présente l'outil d'édition. Cet outil a été créé dans un souci de simplification d'utilisation. Un utilisateur peut via l'éditeur graphique définir son propre EVI et les composants qui le constituent. Cet éditeur se base sur les modèles que nous avons définis pour ensuite générer du code. Ce code correspond aux services proposés par les éléments d'un EVI mais aussi aux interactions entre les différents composants d'un EVI. Ces morceaux de code peuvent ensuite être intégrés à YMCA, l'infrastructure de communication définie dans le chapitre précédent. Ce chapitre permet de conclure la présentation de notre usine logicielle.

Chapitre 7

Conclusion et perspectives

"La science n'atteint jamais son but parce que le but n'en finit pas de se dérober
- et qu'en vérité il n'y a pas de but : la science est une tâche infinie.
Sa grandeur est de se présenter comme un rêve toujours inassouvie."
Jean d'Ormesson, C'est une chose étrange à la fin que le monde

7.1 Résumé des travaux

L'objectif de nos travaux était d'étudier et de définir des solutions pour la génération d'environnements pervasifs dédiés aux espaces de vente. Un état de l'art des projets traitant des projets innovants dans le monde du commerce nous a permis de cerner les limites de ces projets, notamment le manque de méthodologie et d'industrialisation lors de la conception de tels projets. Nous avons alors choisi de développer une usine logicielle basée sur une infrastucture de communication, des modèles et un éditeur graphique.

Afin de mieux appréhender le domaine des espaces de vente intelligents, nous avons opté pour une démarche de co-conception avec les utilisateurs et nous avons suivi une démarche de type *extrem programming* afin de tester rapidement et régulièrement nos propositions.

Dans un premier temps, nous avons travaillé sur la couche de communication. Concernant cette couche de communication, nous avions le choix entre deux approches : utiliser une plate-forme existante ou développer notre propre infrastructure. Nous avons choisi cette seconde approche en nous basant sur les concepts de *pattern*. Nous avons alors réalisé un état de l'art des plates-formes existantes pouvant supporter l'informatique pervasive. Mais aucune solution ne couvre l'ensemble de nos besoins. Par exemple, certaines plates-formes ne prennent pas en compte l'hétérogénéité des éléments pouvant comuniquer. Cette étude nous a permis de mieux définir les besoins technologiques pour notre plate-forme

de communication et ainsi d'identifier les technologies appropriées, OSGi et UPnP. La constitution de l'état de l'art a en effet démontré que l'association des deux technologies répond aux besoins de l'informatique pervasive, notamment en ce qui concerne la gestion du contexte. L'utilisation d'OSGi permet de répondre aux problématiques de l'hétérogénéité car il nous est possible d'implémenter facilement des composants pour créer des liaisons avec d'autres micro-monde technologique. Nous avons alors défini une plate-forme de communication, YMCA, qui possède son propre protocole . Cette plate-forme permet la gestion du contexte, l'appel de services par des périphériques hétérogènes et la gestion du cycle de vie des services (inscription, recherche, ...). Les performances de cette plate-forme ont été étudiées lors de la réalisation de deux prototypes d'assistants à la vente, outils dédiés aux supports pour des vendeurs de grande surface. La plate-forme a parfaitement répondu aux contraintes d'une utilisation réelle, autant en terme de fonctionnalités (gestion des services, prise en compte du contexte,...) qu'en terme de réactivité (temps de réponses , ...). Pour permettre la génération automatique d'espaces de vente intelligents, nous avons dans un second temps défini un ensemble de modèles décrivant les principaux concepts de ces espaces de vente, ainsi que les relations entre les concepts. Ces modèles ont été conçus à partir de l'étude d'espace de ventes, cabine d'essayage augmentée, vitrine intelligente. Cette étude nous a permis de dégager des invariants propres aux EVIs, notamment la description de l'environnement propre aux EVIs(magasins, rayons, ...), les services proposés et les principaux acteurs. Dans un troisième temps, afin de permettre l'industrialisation de la génération des EVIs, nous avons implémenté un éditeur graphique. Cet éditeur se base sur les modèles définis pour générer des morceaux de code. Ces morceaux de code sont ensuite exécutables sur la plate-forme de communication. Enfin, nous avons réalisé un prototype de vitrine augmentée à partir de notre usine logicielle. Cette vitrine a permis de démontrer que les différents éléments de l'usine pouvaient interagir ensemble et que nous savions maitriser la génération d'un EVI depuis la modélisation jusqu'à l'exécution.

7.2 Perspectives

Les perspectives de ce travail portent sur plusieurs points, notamment sur les aspects réalisation, valorisation industrielle et prospectif de recherche. D'un point de vue réalisation, beaucoup de choses sont encore à faire.

Au niveau de la couche de communication, le coeur même de YMCA peut faire l'objet d'un refactoring avec des technologies plus récentes, comme Apache Camel par exemple. L'utilisation de ce framework permettrait d'accélerer les échanges et de permettre une configuration plus fine de la plate-forme.

Au niveau de l'éditeur graphique, les fonctionnalités actuelles sont restreintes. Le but de ces travaux étant une étude de faisabilité, la version de l'éditeur est minimaliste. Par exemple, des outils propres aux éditeurs graphiques doivent être implémentés, l'ergonomie de l'IHM peut être également améliorée.

Au niveau de l'usine logicielle dans sa globalité, il reste aussi à lier les différentes parties entre elles. Le but de l'usine est en effet d'exécuter du code généré par les modèles automatiquement sur la plate-forme YMCA, sans intervention manuelle. De la même manière, une modification effectuée directement sur la plate-forme d'exécution doit aussi être répercutée automatiquement dans les modèles.

L'intégration de mécanismes autour de la sécurité peut aussi être une perspective, notamment dans le cadre de l'intégration dans des SI existants.

L'industrialisation peut aussi être améliorée sur deux points. Le premier concerne les notions de liaisons entre services évoqués dans le chapitre 6. En se basant sur des travaux autour des workflows, en utilisant le langage XPDL par exemple, il serait possible de définir des comportements plus complexes et donc plus intelligents, répondant mieux aux modifications du contexte. Le second point concerne le déploiement de YMCA. Nous pouvons imaginer la génération d'images préconfigurées de la plate-forme et pouvant se personnaliser en fonction de la destination.

D'un point de vue valorisation industrielle, nous avons été approchés par plusieurs entreprises régionales. Ces entreprises ont été interessées par nos prototypes d'EVIs et par la couche de communication YMCA. Cette dernière a pu montrer son efficacité lors du projet p-LearNet mais également lors du salon de la VAD avec le prototype de vitrine augmentée présenté dans le chapitre 6.

D'un point de vue prospectif, nous avons commencé, au cours du projet p-LearNet, une collaboration avec l'équipe de Serge Garlatti, de l'ENST Bretagne, autour de l'informatique pervasive pour l'apprentissage humain. Dans ce cadre, nous avons proposé un modèle de scénariis adaptatif et sensible au contexte [PNGL+09]. Ce modèle permet de définir un workflow dédié à l'apprentissage humain. Les processus décrits par le modèle peuvent s'exécuter sur notre plate-forme YMCA. Nous souhaiterions poursuivre cette collaboration et mener plus en amont les travaux autour de l'apprentissage pervasif.

Nous aimerions également explorer la possibilité offertes par les réseaux de capteurs pour nous fournir des informations plus complètes sur le contexte. L'étude des usages est aussi un point à ne pas négliger pour la suite. De nouveaux dispositifs, comme nous proposons de créer, va entrainer des usages non attendus, en fonction de l'appropriation de ces dispositifs par les utilisateurs. Il faut alors outiller les EVIs et accompagner les expérimentations pour suivre le plus finement possible leur utilisation.

Nous souhaiterions aussi faire évoluer le champ d'application de notre système. Notre usine peut prétendre à des projets orientés domotiques. En utilisant notamment des systèmes embarqués, les Arduino par exemple, il est possible de reproduire un environnement pervasif et de le déployer n'importe où, sous condition d'une couverture réseau correcte. Il faudrait alors créer ou utiliser des modèles existant orientés domotiques pour pouvoir utiliser l'usine logicielle dans sa globalité.

Enfin, nous pensons qu'il serait intéressant de pousser des reflexions autour des interfaces tangibles. Notre plate-forme peut permettre la réalisation rapide de projets intégrants ce genre d'interfaces.

Annexes

Annexe A

Co-conception avec les utilisateurs

FIGURE A.1 – Co-conception des fonctionnalités

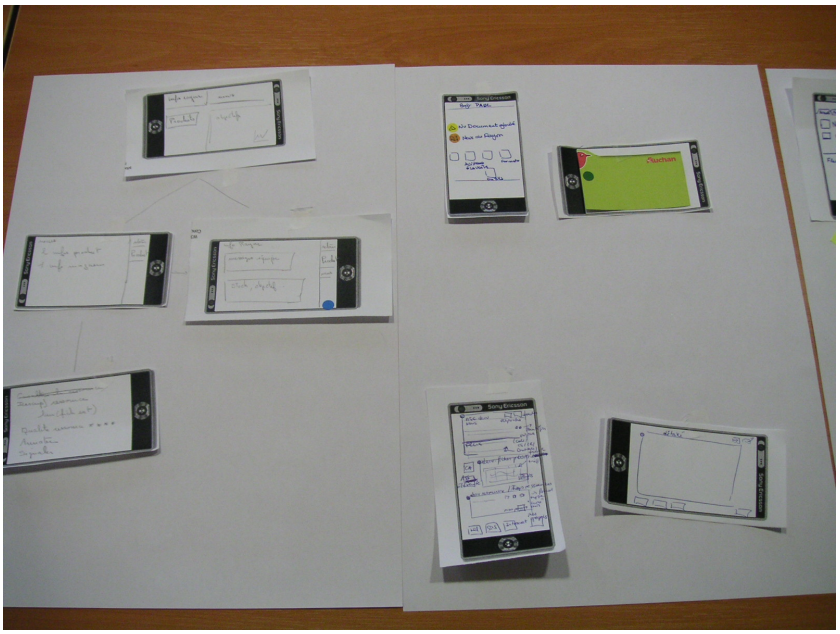

FIGURE A.2 – Co-conception des interfaces utilisateurs

FIGURE A.3 – Co-conception de la cinématique

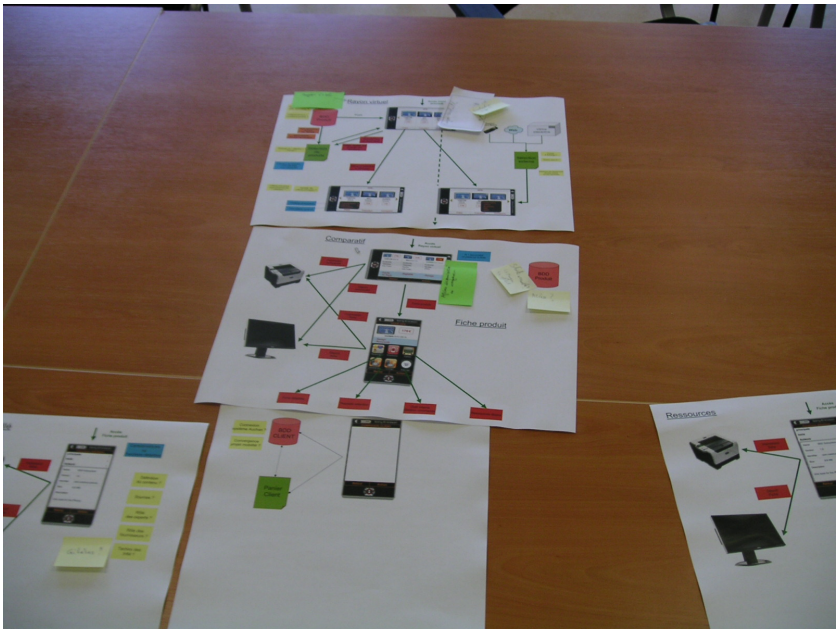

FIGURE A.4 – Deuxième phase de co-conception de la cinématique

Annexe B

Quelques scénarios d'usages

B.1 Scénario vendeur

Michèle assure le métier de conseiller/vendeur pour le rayon Hifi Vidéo d'un hypermarché. Elle vient de commencer son service, un peu avant que le magasin n'ouvre ses portes, et décide de préparer son rayon pour l'ouverture. Son rayon est en réalité un EVI. constitué de trois meubles intelligents, une vitrine interactive, un écran LCD publicitaire permettant d'afficher des informations choisis par Michèle, et une borne interactive. Ces trois éléments peuvent communiquer ensemble : l'écran peut servir de déport d'informations pour la borne et la vitrine, et la vitrine peut s'éclairer si un client cherche sur la borne un produit particulier. Michèle dispose en plus d'un appareil mobile, un PSA [29], appareil que nous détaillerons plus tard. Cet outil lui sert de support à son métier de conseiller/vendeur en lui fournissant des accès à des ressources (fiches produits, ressources explicatives, stocks ...). Mais il lui sert également de télécommande pour piloter l'ensemble des meubles de son rayon.

Avant l'arrivée des clients, elle commence donc par initialiser la publicité affichée à l'écran via son PSA. Elle en profite pour mettre également en état de marche la vitrine et la borne. Ensuite, en consultant les fiches produits, elle vérifie les prix des produits de son rayon.

Une fois ce travail effectué, elle profite de l'absence de client pour parfaire ses connaissances sur une technologie de plus en plus en vogue : le HDMI. Cela lui permettra de répondre plus aisément aux questions fréquentes des clients sur ce sujet. Via l'interface proposée par son PSA, elle décide alors de lancer l'accès aux modules de formation et choisit celui relatif au HDMI. Une fois la séquence terminée, elle décide de compléter sa formation grâce à la FAQ. Au bout de 2-3 questions, elle remarque un client qui est en train de regarder attentivement les écrans plats TV, en particuliers le segment grandes tailles(102 cm et plus). Nous allons maintenant décrire les différentes phases du processus de conseil/vente.

Michèle arrête sa consultation et place son PSA en mode Présence-Client. L'interface s'adapte à ce nouveau contexte : de nouvelles fonctionnalités apparaissent, le comparatif de produits par exemple, d'autres,comme l'accès à la FAQ vendeurs, disparaissent. Michèle commence à dialoguer avec le client

29. Personnal Selling Assistant

afin de mieux cerner ses attentes et le style de client qu'il semble être. Le client souhaite un équipement écran de bonne qualité visuelle, car il est déjà bien équipé du point de vue haute-fidélité. Il décrit son intérêt pour les produits nouveaux dans le domaine de l'imagerie numérique, sa sensibilité à l'innovation...Pour amorcer la démarche d'avant vente, Michèle essaie, par dialogue avec le client, de cerner quelques modèles qui pourraient convenir à son attente. Michèle entre les références de 3 modèles répondant aux attentes du client dans son PSA. Une fois les références rentrées, le système récupère les fiches détaillées de ces produits et construit une synthèse comparative. Cette synthèse sera un tableau récapitulatif des trois modèles sélectionnés, avec des rubriques homogènes. Des informations destinées exclusivement au vendeur apparaissent également (marge, stock...).

Le client semble maintenant un peu plus près d'un choix, mais il se pose encore beaucoup de questions, en particuliers sur les différences, autre que les prix qui sont bien apparents, entre les différents modèles retenus. Pour l'aider à mieux comprendre cela, Michèle l'entraine auprès de l'écran LCD. Par une action simple sur l'écran tactile de son PSA, elle amène la visualisation du tableau comparatif sur l'afficheur. Seules les données pouvant être partagées seront envoyées sur l'afficheur. Arrive notamment l'inévitable question de la comparaison entre technologies PLASMA et LCD. Michèle connait bien se sujet mais préfère illustrer ses propos avec des schémas. Elle recherche alors des ressources correspondant et les envoie sur l'afficheur. En fait, lors de cette courte séance collective d'apprentissage, en juste à temps, elle se retrouve dans la situation d'un tuteur, qui n'a pas à faire appel à sa mémoire pour les éléments d'informations (les petites animations flash qu'elle reconnaît sont d'ailleurs assez auto-explicatives). C'est donc un rôle qui reste valorisant dans sa relation avec le client. Une fois les explications terminées, le client a fait son choix. Le processus de vente reprend son cours normal (remplissage du bon de commande, paiement à la caisse, ...).

B.2 Scénario client

François veut s'acheter un GPS. Il se rend donc sur un site internet dédié et obtient un comparatif de 3 GPS qui correspondent à son budget, ne connaissant pas grand chose aux technologies liées à ces appareils. Il se rend chez un vendeur avec son comparatif afin d'acheter un GPS. Ce magasin a équipé son rayon avec le même ensemble que le scénario précédent.

François n'aime pas perdre son temps dans les magasins et afin de gagner du temps avec le vendeur, il décide de vérifier sur la borne tactile si les produits qu'il a choisis se trouvent bien en rayon. Les trois GPS sont bien dans le rayon mais le moteur de recherche de la borne lui en propose deux autres qui correspondent plus à ses critères, notamment financier. Il décide alors de les ajouter à son comparatif. Là il utilise toujours la borne pour appeler un vendeur. Le PSA de Régis, un vendeur occupé à ranger des appareils en réserve, le prévient qu'un client est en attente devant la borne. Régis décide d'aller à sa rencontre. François lui présente le comparatif. En reconnaissant les produits, Régis peut les scanner pour reproduire sur son appareil un comparatif, mais plus détaillé. Il emmène alors François devant un écran LCD du rayon servant de PLV [30] et lui affiche le comparatif. À partir de là, Régis commence son

30. Promotion sur le Lieu de Vente

argumentaire de vente en essayant de capter les besoins de François et surtout son budget. François veut un GPS simple d'utilisation, avec un écran large, et donnant des informations sur le trafic en temps réel. Régis trouve alors un produit qui correspond plus aux attentes de François. Il lui suffit de scanner ce produit pour l'ajouter au comparatif en cours. François est intéressé et demande plus de détails concernant ce GPS. Régis lui affiche alors la fiche détaillée du produit. François compte acheter ce produit. Mais soudain, il se demande comment se passe la mise à jour des cartes. Régis retrouve rapidement dans sa banque de ressources une petite animation explicative et l'affiche sur l'écran. L'achat peut donc se conclure. Régis bascule l'écran en mode "publicité" et édite le bon d'achat en remplissant le formulaire sur son dispositif. Il imprime ensuite le bon via l'imprimante bluetooth présente dans son rayon.

B.3 Scénario créateur de services

Newave est une entreprise fournissant des environnements intelligents pour les lieux de vente. Aujourd'hui, elle a reçu une nouvelle demande provenant des supermarchés Zirgo. Ils souhaitent mettre en place un ensemble de meubles intelligents visant à doper leurs ventes. Leur stratégie commerciale consiste à commencer par leur secteur hifi-video. Cet ensemble est constitué d'un nouveau dispositif d'affichage, d'une borne interactive tactile devant chaque rayon, et d'un système de repérage pour retrouver un produit particulier, piloté depuis la borne.

Roger, ingénieur chez Newave, doit s'occuper de cette demande. Il se connecte sur son logiciel, qui est une usine logicielle spécialisée dans la création d'Espaces de vente. Ce logiciel propose une interface simple pour générer les éléments nécessaires à la création de l'ensemble : commande des meubles auprès des fournisseurs, génération des briques logiciels correspondant aux interactions demandées, conseils pour l'installation, ...À partir de cet outil, il pourra envoyer plusieurs devis.

La première étape est la création d'un nouveau projet, appelé 'TrouveProduitForZirgo' pour l'occasion. Il recherche alors les différents meubles identifiés : borne interactive et écran. À partir d'une recherche basé sur le service de repérage d'un produit, le logiciel lui propose des systèmes à base de video-projecteur monté sur un moteur pour pouvoir éclairer différentes zones. Roger configure tout cela et crée les services associés à l'ensemble : service de définition des coordonnées spatiales des produit pour le moteur, éclairage ou non. Ensuite, il s'attaque à la borne. La borne se connecte aux SI de Zirgo et doit permettre une recherche de produits. Elle doit ensuite envoyer au vidéo-projecteur les coordonnées associées au produit pour pouvoir l'éclairer.

Roger représente graphiquement un lien entre le vidéo et la borne et ensuite configure le lien : il envoie les coordonnées qui seront récupérées par le service affichage du vidéo. Ce logiciel génère ensuite des squelettes de code qu'il faudra compléter pour lier le système et le SI.

Annexe C

Schéma XSD de description d'un fournisseur de services

```
<?xml version "1.0" encoding "UTF-8"?>
<xs:schema       xs "http://www.w3.org/2001/XMLSchema">
 <xs:element name "var">
  <xs:complexType>
   <xs:attribute name "name" use "required">
    <xs:simpleType>
     <xs:restriction base "xs:string">
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
  </xs:complexType>
 </xs:element>
 <xs:element name "statevar">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref "var" maxOccurs "unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name "services">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref "service"/>
   </xs:sequence>
  </xs:complexType>
```

```
26 </xs:element>
27 <xs:element name "service">
28  <xs:complexType>
29   <xs:sequence>
30    <xs:element ref "statevar"/>
31    <xs:element ref "actions"/>
32   </xs:sequence>
33   <xs:attribute name "name" use "required">
34    <xs:simpleType>
35     <xs:restriction base "xs:string">
36
37     </xs:restriction>
38    </xs:simpleType>
39   </xs:attribute>
40  </xs:complexType>
41 </xs:element>
42 <xs:element name "retour">
43  <xs:complexType>
44   <xs:attribute name "type" use "required">
45    <xs:simpleType>
46     <xs:restriction base "xs:string">
47           </xs:restriction>
48    </xs:simpleType>
49   </xs:attribute>
50  </xs:complexType>
51 </xs:element>
52 <xs:element name "param">
53  <xs:complexType>
54   <xs:attribute name "type" use "required">
55    <xs:simpleType>
56     <xs:restriction base "xs:string">
57     </xs:restriction>
58    </xs:simpleType>
59   </xs:attribute>
60   <xs:attribute name "name" use "required">
61    <xs:simpleType>
62     <xs:restriction base "xs:string">
63     </xs:restriction>
64    </xs:simpleType>
65   </xs:attribute>
```

```
66   </xs:complexType>
67  </xs:element>
68  <xs:element name "methode">
69   <xs:complexType>
70    <xs:attribute name "name" use "required">
71     <xs:simpleType>
72      <xs:restriction base "xs:string">
73      </xs:restriction>
74     </xs:simpleType>
75    </xs:attribute>
76   </xs:complexType>
77  </xs:element>
78  <xs:element name "fournisseur">
79   <xs:complexType>
80    <xs:sequence>
81     <xs:element ref "services"/>
82    </xs:sequence>
83    <xs:attribute name "typeFournisseur" use "required">
84     <xs:simpleType>
85      <xs:restriction base "xs:string">
86      </xs:restriction>
87     </xs:simpleType>
88    </xs:attribute>
89    <xs:attribute name "protocole" use "required">
90     <xs:simpleType>
91      <xs:restriction base "xs:string">
92      </xs:restriction>
93     </xs:simpleType>
94    </xs:attribute>
95    <xs:attribute name "name" use "required">
96     <xs:simpleType>
97      <xs:restriction base "xs:string">
98      </xs:restriction>
99     </xs:simpleType>
100   </xs:attribute>
101  </xs:complexType>
102 </xs:element>
103 <xs:element name "actions">
104  <xs:complexType>
105   <xs:sequence>
```

```
106     <xs:element ref "action"/>
107    </xs:sequence>
108   </xs:complexType>
109  </xs:element>
110  <xs:element name "action">
111   <xs:complexType>
112    <xs:sequence>
113     <xs:element ref "methode"/>
114     <xs:element ref "param"/>
115     <xs:element ref "retour"/>
116    </xs:sequence>
117    <xs:attribute name "name" use "required">
118     <xs:simpleType>
119      <xs:restriction base "xs:string">
120      </xs:restriction>
121     </xs:simpleType>
122    </xs:attribute>
123   </xs:complexType>
124  </xs:element>
125 </xs:schema>
```

Index

Bibliographie

[ADB+99] G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Handheld and Ubiquitous Computing*, pages 304–307. Springer, 1999.

[ALI] `http://www.alice.org/`.

[apa] `http://camel.apache.org/`.

[AUR] `http://www.cs.cmu.edu/~aura/`.

[Bar05] E. Bardram. Activity-based computing: support for mobility and collaboration in ubiquitous computing. *Personal and Ubiquitous Computing*, 9(5):312–322, 2005.

[BBC97] P.J. Brown, J.D. Bovey, and X. Chen. Context-aware applications: from the laboratory to the marketplace. *Personal Communications, IEEE*, 4(5):58–64, 1997.

[BBMW02] Jean-Louis Bénard, Laurent Bossavit, Régis Médina, and Dominic Williams. *L'eXtrem Programming*. Eyrolles, mai 2002. isbn=2-212-11051-0.

[BC01] G. Bieber and J. Carpenter. Introduction to service-oriented programming (rev 2.1). *OpenWings Whitepaper, April*, 2001.

[BCO+02] F. Barbier, C. Cauvet, M. Oussalah, D. Rieu, S. Bennasri, and C. Souveyet. Composants dans l'ingénierie des systèmes d'information: concepts clés et techniques de réutilisation. *Actes des deuxièmes assises nationales du GDR-I3*, 2002.

[Bec02] Kent Beck. *eXtrem Programming*. Campus Press, 2002. isbn=2-7440-1433-8.

[BLM02] M. Beaudouin-Lafon and W. Mackay. Prototyping development and tools. *Handbook of Human-Computer Interaction. New York: Lawrence Erlbaum Associates*, pages 1006–1031, 2002. LNCS 4500.

[bpe] `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`.

[bpm] `http://www.bpmn.org/`.

[CAG05] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927 (Proposed Standard), May 2005.

[cam] `http://camel.apache.org/`.

[CC02] G. Calvary and J. Coutaz. Plasticité des interfaces: une nécessité. *information-interaction-intelligence, Actes des deuxièmes Assises nationales du GDR I*, 3:247–261, 2002.

[CD04] R. Chalon and B.T. David. Modélisation de l'interaction collaborative dans les systèmes de réalité mixte. In *Proceedings of the 16th conference on Association Francophone d'Interaction Homme-Machine*, pages 37–44. ACM, 2004.

[CDR06] V. Chevrin, A. Derycke, and J. Rouillard. Project ubi-learn: An intermediation infrastructure for multi-channel accesses to future lms. In *Telecommunications, 2006. AICT-ICIW'06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pages 7–7. IEEE, 2006.

[Cha04] D.A. Chappell. *Enterprise service bus*. O'reilly Media, 2004.

[Che06] Vincent Chevrin. *L'interaction usagers/services, multimodale et multicanale : une première proposition appliquée au domaine du e-commerce*. PhD thesis, Université Lille 1 - Sciences et Technologies, avril 2006.

[clo] `http://www.syntec-numerique.fr/Actualites/Publication-du-Livre-Blanc-Cloud-Computing-de-`

[DA00] A.K. Dey and G.D. Abowd. The context toolkit: Aiding the development of context-aware applications. In *Workshop on Software Engineering for wearable and pervasive computing*, pages 431–441, 2000.

[DCV07] Alain Derycke, Vincent Chevrin, and Thomas Vantroys. P-learning and e-retail: a case study and a flexible software architecture, 2007. In Pervasive Learning 2007 workshop of the Pervasive 2007 conference, Toronto, Canada, 13-16 may 2007.

[Der08] Alain Derycke. Apprentissage et intelligence ambiante. Keynote, ConfÃ©rence TICE 2008, octobre 2008.

[DH00] P. Den Hertog. Knowledge-intensive business services as co-producers of innovation. *International Journal of Innovation Management*, 4(4):491–528, 2000.

[Dou01] P. Dourish. Seeking a foundation for context-aware computing. *Human-Computer Interaction*, 16(2):229–241, 2001.

[DVBL08] Alain Derycke, Thomas Vantroys, Benjamin Barbry, and Philippe Laporte. E-retail: Interaction of intelligent selling space with personal selling assistant. Barcelona, Spain, 2008. ICEIS.

[ecr] `http://www.ecr-all.org/`.

[Eng62] Douglas Engelbart. Augmenting human intellect: a conceptual framework. Technical Report AFOSR-3223, Stanford Research Institute, october 1962. `http://www.dougengelbart.org/pubs/augment-3906.html`.

[Eng68] Douglas Engelbart. Bootstrapping approach, 1968. `http://www.sri.com/news/storykits/videos/Bootstrapping.mov`.

[Eur01] E.C.R. Europe. *A Guide to CPFR implementation*. ECR Europe Publ. Office, 2001.

[eva] `http://evalab.univ-lille2.fr/`.

[evr] `http://www.ma-residence.fr/`.

[Far07] Farmer and Lindstaedt and Droschl and Luttenberger. AD-HOC - Work-integrated Technology-supported Teaching and Learning. In *Proceedings of the 5th International Conference on Organizational Knowledge, Learning, and Capabilities, Innsbruc*, volume 12, pages 117–156, 2007.

[FJRT07] I.R. Floyd, M.C. Jones, D. Rathi, and M.B. Twidale. Web mash-ups and patchwork prototyping: User-driven technological innovation with web 2.0 and open source software. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pages 86–86. Ieee, 2007.

[fle] `http://www.adobe.com/fr/products/flex.html`.

[FM07] J. Füller and K. Matzler. Virtual product experience and customer participation-a chance for customer-centred, really new products. *Technovation*, 27(6):378–387, 2007.

[Gam95] E. Gamma. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.

[GBS03] P. Grace, G. Blair, and S. Samuel. Remmoc: A reflective middleware to support mobile client interoperability. *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 1170–1187, 2003.

[ges] `www.gesturetek.com`.

[GS04] Jack Greenfield and Keith Short. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Wiley, 2004. ISBN: 0-471-20284-3.

[GTPL09] D. Guinard, V. Trifa, T. Pham, and O. Liechti. Towards physical mashups in the web of things. In *Networked Sensing Systems (INSS), 2009 Sixth International Conference on*, pages 1–4. IEEE, 2009.

[Hur98] J. Hurwitz. Sorting out middleware. *DBMS*, 11(1):10–12, 1998.

[Jay08] L. Jayr. *Flex 3: Applications Internet riches*. Editions Eyrolles, 2008.

[KBBC05] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza. Tuio: A protocol for table-top tangible user interfaces. In *Proc. of the The 6th Int'l Workshop on Gesture in Human-Computer Interaction and Simulation*, 2005.

[KOO08] S. Keegan, G.M.P. O'Hare, and M.J. O'Grady. Easishop: Ambient intelligence assists everyday shopping. *Information Sciences*, 178(3):588–611, 2008.

[Kou03] G. Kourouthanassis, P. Roussos. Developping consumer-friendly pervasive retail systems. *Pervasive computing, IEEE*, 2(2):32–39, 2003.

[KR07] S. Konomi and G. Roussos. Ubiquitous computing in the real world: lessons learnt from large scale rfid deployments. *Personal and Ubiquitous Computing*, 11(7):507–521, 2007.

[Kum09] Adam Kumpf. Trackmate: Large-scale accessibility of tangible user interfaces. Master's thesis, Massachusetts Institute of Technologuy, June 2009.

[liga] `http://blog.xebia.fr/2007/12/17/spring-integration-lavenement-des-lightweight-esb/`.

[ligb] http://www.codeproject.com/Articles/99090/Implementing-a-Lightweight-ESB-Using-Content-Based.

[LLSW01] R. Levine, C. Locke, D. Searls, and D. Weinberger. *The cluetrain manifesto: the end of business as usual*. Perseus Pub., 2001.

[LTE] `http://www.3gpp.org/LTE`.

[LV01] R. Lefébure and G. Venturi. *Gestion de la relation client*. Eyrolles, 2001.

[LW06] Gunther Lenz and Christoph Wienands. *Practical Software Factories in .NET*. Apress, 2006. ISBN: 978-1-59059-665-4.

[Mac96] W.E. Mackay. Réalité augmentée: le meilleur des deux mondes. *Recherche*, (285):32–37, 1996.

[min] `mindstorms.lego.com/`.

[mir] `http://dvice.com/archives/2010/02/japanese-augmen.php`.

[ML99] Stéphane Gasch Marc Langlois. *Le commerce électronique B to B : de l'EDI à Internet*. 1999.

[mul] `www.mulesoft.com/`.

[NFC] `http://www.nfc-forum.org/`.

[NRB68] P. Naur, B. Randell, and F.L. Bauer. *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October, 1968*. Scientific Affairs Division, NATO, 1968.

[Org10] International Standards Organisation. Iso 9241-210 - ergonomie de l'interaction homme-système – partie 210: Conception centrée sur l'opérateur humain pour les systèmes interactifs, 2010. révision de la norme ISO 13407:1999.

[PBV+12] Y. Peter, B. Barbry, T. Vantroys, P. Laporte, and S. Lerouge. Design and evaluation of a pervasive workplace learning system for retail stores. In *Advanced Learning Technologies (ICALT), 2012 IEEE 12th International Conference on*, pages 202–204. IEEE, 2012.

[PNGL+09] C. Pham-Nguyen, S. Garlatti, B.Y. Lau, B. Barbry, T. Vantroys, et al. An adaptive and context-aware scenario model based on a web service architecture for pervasive learning systems. *International Journal of Mobile and Blended Learning (IJMBL)*, 1(3):41–69, 2009.

[PTDL07] M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45, 2007.

[Qui] H.L.G.G.R. Quinn. Prada epicenter, new york. `http://archtlas.com/sites/default/files/obras/prada-epicenter-soho/archivos/prada.pdf`.

[RD08] Tijs Rademakers and Jos Dirksen. *Open Source ESBs in Action*. Manning Publications, octobre 2008.

[RPM97] N. Ryan, J. Pascoe, and DR Morse. Enhanced reality fieldwork: the context-aware archaeologist assistant. *Archaeology in the age of the Internet: Computer Applications & Quantitative Methods in Archaeology*, pages 269–274, 1997.

[RTK+02] George Roussos, Juha Tuominen, Leda Koukara, Olli Seppala, Panos Kourouthanasis, George Giaglis, and Jeroen Frissaer. A case study in pervasive retail. In *Proceedings of the 2nd international workshop on Mobile commerce*, WMC '02, pages 90–94, New York, NY, USA, 2002. ACM.

[SAW94] BN SHILIT, N. Adams, and N. WANT. Context-aware computing applications. *Proceedings of the IEEE Workshho on Mobile Computing System and Application*, pages 85–90, 1994.

[SBP+08] J.P. Sousa, R.K. Balan, V. Poladian, D. Garlan, and M. Satyanarayanan. User guidance of resource-adaptive systems. 3rd International Conference on Software and Data Technologies (ICSOFT), 2008.

[SCR] `http://scratch.mit.edu/`.

[sera] `http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf`.

[serb] `http://servicemix.apache.org`.

[spa] `http://www.w3.org/TR/rdf-sparql-query/`.

[Spé] `http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications`.

[SSM04] J. Struker, S. Sackmann, and G. Muller. Case study on retail customer communication applying ubiquitous computing. In *e-Commerce Technology, 2004. CEC 2004. Proceedings. IEEE International Conference on*, pages 42–48. IEEE, 2004.

[STV05] M. Sharples, J. Taylor, and G. Vavoula. Towards a theory of mobile learning. In H. van der Merwe and T. Brown, editors, *Proceedings of mLearn 2005 Conference*, pages 1–9. University of Birmingham, 25-28 october 2005.

[TC02] D. Thevenin and J. Coutaz. Adaptation des ihm: taxonomies et archi. logicielle. In *Proceedings of the 14th French-speaking conference on Human-computer interaction (Conférence Francophone sur l'Interaction Homme-Machine)*, pages 207–210. ACM, 2002.

[Tri99] C. Trigueros. Albatros: Electronic tagging solutions for the retail sector. *Informatica El Corte Inglés*, 1999.

[ubi] `www.ubiquarium.fr/`.

[VL05] G. Vanwormhoudt and E.T. Lille. Précision et validation de métamodèles avec emf et ocl. *OCM 2005*, page 19, 2005.

[VMS+06] Giasemi N. Vavoula, Julia Meek, Mike Sharples, Peter Lonsdale, and Paul Rudman. A lifecycle approach to evaluating myartspace. In *WMTE*, pages 18–22. IEEE Computer Society, 2006.

[VSR+09] Giasemi Vavoula, Mike Sharples, Paul Rudman, Julia Meek, and Peter Lonsdale. Myartspace: Design and evaluation of support for learning with multimedia phones between classrooms and museums. *Comput. Educ.*, 53:286–299, September 2009.

[WCO11] DerniÃ¨re visite 17 mars 2011. `https://www.wcomp.fr`.

[Wei93a] M. Weiser. Ubiquitous computing. *Computer*, 26:71–72, October 1993.

[Wei93b] Mark Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, 36:75–84, July 1993.

[Wei99] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3:3–11, July 1999.

[WGB99] M. Weiser, R. Gold, and J. S. Brown. The origins of ubiquitous computing research at parc in the late 1980s. *IBM Syst. J.*, 38:693–696, December 1999.

[wik] `http://www.wikitude.com/`.

[WW06] Rainer Wasinger and Wolfgang Wahlster. Multi-modal human-environment interaction. In Emile Aarts and José Luis Encarnação, editors, *True Visions: The Emergence of Ambient Intelligence*, pages 291–306. Springer, Berlin, Heidelberg, New York, 2006.

[YMKM00] J. Youll, J. Morris, R. Krikorian, and P. Maes. Impulse: Location-based agent assistance. In *Software Demos, in Proc. of the Fourth International Conference on Autonomous Agents*. Citeseer, 2000.

[zig] `http://www.zigbee.org/`.

Résumé

Dans cette thèse, nous explorons les concepts de l'intelligence ambiante et les mécanismes facilitant leurs usages pour la conception d'environnements de vente intelligents. Une partie du travail fût réalisée dans le cadre du projet ANR p-LearNet. Notre objectif est de démontrer la faisabilité d'une solution permettant la génération d'espaces de ventes intelligents. Nous avons alors réalisé une usine logicielle. Cette usine a pour objectif, à partir de modèles, de générer tout ou partie de code permettant l'exécution de l'espace de vente. Ce code peut être exécuté sur une plate-forme de communication intégrée à l'usine. Nous avons dans un premier temps défini une infrastructure de communication permettant de gérer un ensemble de dispositifs hétérogènes. Cette plate-forme est un ESB léger dédié aux espaces de ventes intelligents. Cette plate-forme a été testée au cours du projet p-LearNet. Dans un second temps, nous avons créé des modèles qui représentent notre domaine d'application, les espaces de vente intelligents. Ces modèles permettent de définir les principaux éléments qui vont constituer un espace intelligent : les acteurs, les meubles, les services proposés, ...À partir de ces modèles, nous avons implémenté un éditeur graphique. Cet éditeur est à destination d'un utilisateur non-informaticien pour qu'il puisse graphiquement construire un espace de vente intelligent. Cet outil génère ensuite du code qui peut être exécuté sur la plate-forme décrite précédemment.

Mots-clés: Informatique pervasive, infrastructure logicielle,logiciel médiateur, usine logicielle, e-retail

Abstract

In this thesis, we explore the concepts of ambient intelligence and mechanisms to facilitate their use for designing intelligent selling environments. Part of the work was carried out within the ANR p-LearNet project. Our goal is to demonstrate the feasibility of a solution to the generation of intelligent selling spaces. Then we performe a software factory. This software factory aims to generate all or part of code to run the selling space from models. This code can be run on a communication platform integrated within the factory. We initially defined a communication infrastructure to manage a set of heterogeneous devices. This platform, based on OSGi and UPnP, is considered as a lightweight ESB dedicated to intelligent selling spaces. This platform has been tested during p-LearNet project and deployed within supermarkets. In a second step, we have created models that represent our scope, intelligent selling spaces. These models define the key elements that will form a smart space: actors, furniture, services offered, ...From these models, we have implemented a graphical editor. This editor is intended for a user, of which knowledge in computer sciences is poor, to graphically build an intelligent selling space. Then the tool generates code that can be executed on the platform described above. The use of this tool has been demonstrated through the implementation of an improved display casewindow. This showcase, presented at the VAD Lille salon, is running on the communication platform defined in the software factory.

Keywords: Pervasive computing, software infrastructure, middleware, software factory, e-retail