وزارة التعليم العالي و البحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار - عنابة

Faculté des sciences de l'ingénieur **Année :** 2012

Département d'informatique

MEMOIRE

Présenté en vue de l'obtention du diplôme de **MAGISTER**

Une Architecture Agents pour l'Adaptation au Contexte des Systèmes d'Information Ubiquitaires

Option

Génie logiciel

Par

Sonia Achiri

DIRECTEUR DE MEMOIRE : Habiba BELLEILI MCA Université Badji Mokhtar Annaba

DEVANT LE JURY

PRESIDENT: Mr Mohammed Tahar KIMOUR PRF Université Badji Mokhtar Annaba

EXAMINATEURS: Melle Fadila ATIL MCA Université Badji Mokhtar Annaba

Mme Nora BOUNOUR MCA Université Badji Mokhtar Annaba

ملخـص

يندرج هذا العمل في الإعلام الآلي المحيط ، وهو ميدان جديد للبحث، تحت عنوان " une architecture agents pour l'adaptation au contexte des Systèmes d'Information Ubiquitaires. ".

إن التطور التكنولوجي لنظم الكمبيوتر والاتصالات اليوم دفع مصممي البرامج لدمج الأجهزة النقالة في تطبيقاتهم. في نظام المعلومات المحيط، يحتاج التطبيق إلى التكيف مع سياقات مختلفة حتى يوفر للمستخدم الخدمة المناسبة والبيانات المفيدة في شكل يوافق رغباته.

الهدف الرئيسي لهذه الم ذكرة هو اقتراح نموذ ج لتمثيل العوامل الخارجية للنظام (مكان المستعمل، رغباته، نوع الجهاز...). كأداة اخترنا استخدام XML.

استعملنا مفهوم "المعلومات التي تؤدي إلى بدء خدمة والمعلومات التي تغير شكل الخدمة" في عملية تصميم التطبيقات الحساسة للظروف الخارجية ، لضمان تكييف خدمات التطبيق.

نفذنا دراسة على أساس مشروع "مساعد المؤتمر" وهو بحث لداي وآخرين [Dey et al., 1999] مثلنا العوامل الخارجية للنظام وفقا لنموذجنا، و قمنا بتصميم التطبيق للتحقق من اقتراحنا لضبط و تكييف الخدمة.

كلمات المفاتيح: العوامل الخارجية للنظام ، *تكييف الأنظمة*، الإعلام الآلي المحيط ، رغبات المستعمل.

Résumé

De nos jours, l'évolution technologique des systèmes embarqués et des moyens de communication informatiques a incité les développeurs à intégrer les terminaux mobiles dans leurs applications. Dans un système d'information pervasif, l'application doit s'adapter aux différents contextes pour offrir à l'utilisateur le service approprié ainsi que les données utiles dans un format approprié.

Dans notre travail, nous proposons un modèle du contexte dans les systèmes d'information pervasifs (SIP), en partant d'autres définitions de la littérature. Comme outil de modélisation du contexte, nous avons choisi d'utiliser XML.

Nous introduisons la notion d'informations qui déclenchent un service et des informations qui changent la forme d'un service, dans le processus de conception d'applications sensibles au contexte, pour assurer l'adaptation du comportement de l'application.

Nous avons implémenté un cas d'étude, tiré d'un projet de recherche « Conférence Assistant » de Dey, Salber et Abow [Dey et al., 1999]. Nous avons représenté les informations contextuelles de ce projet selon notre modèle et nous avons conçu l'application pour valider notre proposition d'adaptation de service.

Mots clés : *Systèmes d'information ubiquitaires, préférences de l'utilisateur, contexte d'utilisation, adaptation, modélisation du contexte, Systèmes pervasifs.*

Abstract

Today, the technological evolution of embedded systems and computer communication has prompted developers to integrate mobile devices into their applications. In a pervasive information system, the application needs to adapt to different contexts to provide the user with the appropriate service and data in a useful format.

In our work, we propose a model of the context in pervasive information systems (SIP), from other definitions of literature. As a tool for modeling context, we have chosen to use XML

We introduce the notion of information that trigger a service and information that change the form of a service in the design process of context-aware applications, to ensure the adaptation of application behavior.

We have implemented a case study, based on a research project "Conference Assistant" of Dey, Salber and Abow [Dey et al., 1999]. We represented the contextual information of this project according to our model and we designed the application to validate our proposal for adaptation of services.

Keywords: Ubiquitous computing, Context-awareness, *adaptation, SMA, preferences of user*

Table des matières

ملخـص i

Résumé ii

Abstract iii

Table des matières V

Liste des tableaux Vi

Liste des figures Viii

Introduction générale

1. Problématique 2

2. Contribution 2

3. Plan du mémoire 3

CHAPITRE I Modélisation du contexte des systèmes d'information pervasifs 5

I.1 Exemple illustratif 6

I.2. Présentation des systèmes d'information pervasifs 8

I.3. Contexte et informatique pervasive 12

I.3.1. Définitions du contexte 12

I.3.2. Contexte pertinent 14

I.3.3. Utilisation du contexte 15

I.3.4. Catégories de contexte 15

I.4. Notion de la sensibilité au contexte (Context-Awareness) 17

I.5. Modélisation du contexte 18

I.5.1. Les modèles Attribut/Valeur 18

I.5.2. Les modèles de représentation par balises 19

I.5.3. Les modèles graphiques 21

I.5.4. Les modèles orientés objets 22

I.5.5. Les modèles logiques 23

I.5.6. Les modèles basés sur les ontologies 24

Conclusion 28

CHAPITRE II Architectures des systèmes sensibles au contexte 29

II.1. Architecture d'un système sensible au contexte 30

II.1.1. Couche de capture du contexte 32
II.1.2. Couche d'Interprétation et d'agrégation du contexte 33
II.1.3. Couche de stockage et historique du contexte 33
II.1.4. Couche dissémination du contexte 34
II.1.5. Couche application 34
II.2. Quelques architectures des systèmes sensibles au contexte 35
II.2.1 ActiveBadge (1992) 35
II.2.2 ParcTab (1995) 37
II.2.3 Stick-e-notes (1997) 38
II.2.4 Cyberguide (1997) 39
II.2.5 Context Toolkit (2001) 39
II.2.6 Context management framework (2003) 40
II.2.7 Hydrogen (2003) 42
II.2.8 CASS (2004) 43
II.2.9 CORTEX (2004) 44
II.2.10 SOCAM (2004) 45
II.2.11 CoBrA (2004) 46
II.2.12 JCAF (2005) 47
II.2.13 Analyse des architectures 49
II.3. Les caractéristiques souhaitées des applications sensibles au contexte 50
II.4. Les caractéristiques fondamentales des systèmes d'information pervasifs 50
II.5. L'adaptation 51
II.5.1. Définition de l'adaptation 51
II.5.2. Classification de l'adaptation 52
II.5.3. Adaptation dans les systèmes pervasifs 55
II.5.3.1. Adaptation de contenu multimédia 55
II.5.3.2. Adaptation de services 57
II.5.4. Le processus d'adaptation 58
Conclusion 58

CHAPITRE III : La conception 59
III.1. Paramètres d'adaptation 60

III.2. Travaux similaires 61
III.2.1. Définition et modélisation du contexte 61
III.2.2. Adaptation au contexte 63
III.2.3. Gestion des préférences de l'utilisateur 64
III.3. Présentation générale du modèle du contexte proposé 65
III.4. Paramètres de l'utilisateur 68
III.4.1. Préférences d'activité 70
III.4.2. Préférences de résultat 70
III.4.3. Préférences d'affichage 70
III.5. Adaptation des applications sensibles au contexte dans les systèmes pervasifs 71
Conclusion 73

CHAPITRE IV: Implémentation 74
IV.1. Réalisation d'une application sensible au contexte 75
IV.1.1. Les caractéristiques du PDA 76
IV.2. Description de l'application 78
IV.3. Conception du logiciel « Assistant de conférence » 79
IV.3.1. Processus de gestion du contexte 79
IV.3.2. Spécification des services de l'application 80
IV.3.3. Représentation des informations du contexte 81
IV.3.4. Gestion des préférences de l'utilisateur 83
IV.4. Les outils du développement 85
IV.5. Implantation du logiciel 86
Conclusion 91
V. Conclusion et Perspectives 93
V.1. Conclusion 94
V.2. Perspectives 95
Références bibliographiques 96
Références bibliographiques 97

Liste des tableaux

Tableau IV.1. Les services de l’application. 86

Liste des figures

Figure I.1: Exemple tiré du scénario du projet européen GLOSS. 7

Figure I.2: Évolution du marché mondial des ordinateurs. Adaptée de [Riveill, 2002] 8

Figure I.3: Évolution vers l'informatique pervasive Adaptée de [Satyanarayanan, 2001] 9

Figure I.4: Les composants d'un système pervasif. Adaptée de [Saha et Mukherjee, 2003] 11

Figure I.5: Exemple un message ConteXtML. 19

Figure I.6: Modélisation de la localisation de l'utilisateur en utilisant CC/PP 20

Figure I.7: L'ontologie à deux niveaux de l'approche CONON. Adaptée de [Wang et al., 2004] 26

Figure II.1: Architecture générale d'un système sensible au contexte. Adaptée de [chaari, 2007] 32

Figure II.2: Architecture de l'ActiveBadge. Adaptée de [Want et al., 1992] 36

Figure II.3: Architecture du ParcTab. Adaptée de [Want et al., 1995] 38

Figure II.4: Éléments de l'architecture du Contexte Toolkit. Adaptée de [Dey et al., 2001] 40

Figure II.5: Architecture de CMF. Adaptée de [Korpipää et al., 2003] 42

Figure II.6: Architecture de Hydrogen. Adaptée de [Hofer et al., 2003] 42

Figure II.7: Architecture de CASS. Adaptée de [Fahy et Clarke, 2004] 43

Figure II.8: Architecture d'un « sentient object ». Adaptée de [Biegel et Cahill, 2004] 44

Figure II.9: Architecture de SOCAM. Adaptée de [Gu et al., 2004] 46

Figure II.10: Architecture de CoBrA. Adaptée de [Chen, 2004] 47

Figure II.11: Architecture de JCAF. Adaptée de [Bardram, 2005] 48

Figure III.1. Les éléments de l'ontologie de service de Miraoui. 62

Figure III.2. Le modèle général du contexte de Chaari. 63

Figure III.3. Les composants d'un système pervasif. 63

Figure III.4. Diagramme de classes de la représentation du contexte. 65

Figure III.5. Caractéristiques du dispositif. 67

Figure III.3. Caractéristiques de l'utilisateur. 68

Figure III.4. Processus de gestion du contexte 72

Figure IV.1. Planning des présentations de la conférence. 88

Figure IV.2. Planning de la conférence avec les informations sur les présentations. 88

Figure IV.3. Création du participant. 89

Figure IV.4. Planning avec les niveaux d'intérêts. 89

Figure IV.5. Planning des sessions intéressantes selon les centres d’intérêt de l’utilisateur. 90

Figure IV.6. L'interface de prise de notes. 90

Introduction générale

1. Problématique

Ce travail de recherche s'inscrit dans le domaine naissant de l'informatique pervasive avec, comme sujet d'étude, une architecture agents pour l'adaptation au contexte des systèmes d'information ubiquitaires.

L'informatique ubiquitaire, que l'on décline sous différents termes, correspond à cette (r)évolution technique envisagée il y a une quinzaine d'années par Weiser [Weiser, 1991]. Par opposition à l'informatique traditionnelle, la nouveauté tient aux capacités de mobilité et d'intégration des systèmes dans le milieu physique [Lyytinen et Yoo, 2002], et ceci de manière spontanée et à de multiples échelles.

Les systèmes pervasifs ont pour objectif de rendre l'information disponible partout et à tout moment. Ces systèmes doivent pouvoir être utilisés dans différents contextes selon l'environnement de l'utilisateur, son profil et le terminal qu'il utilise. L'un des problèmes majeurs de ce type de systèmes concerne donc l'adaptation au contexte d'utilisation.

Suite à notre recherche dans le domaine de la sensibilité au contexte, la grande majorité des contributions existantes dans ce domaine s'intéressent à comment capturer, interpréter et apporter les informations contextuelles à l'application sans étudier comment l'adapter à ces nouveaux contextes d'utilisation.

D'autre part, la notion du contexte reste vague et les travaux existants proposent soit des définitions abstraites soit ils énumèrent certains éléments du contexte selon leurs champs d'application. Ainsi, plusieurs modèles du contexte ont été proposés pour diverses finalités.

2. Contribution

Le problème qui se pose pour les systèmes pervasifs est que le concepteur ne peut être conscient dès le départ de toutes les informations qui influencent sur le comportement de l'application. C'est pourquoi, nous proposons un modèle générique et évolutif basé sur celui proposé par Chaari [Chaari, 2007], combiné avec la définition orientée service de Miraoui [Miraoui, 2009].

Nous introduisons dans ce modèle de contexte la notion de préférences de l'utilisateur ainsi que l'aspect des caractéristiques dynamiques.

La notion des informations qui déclenchent un service et des informations qui changent la forme d'un service, nous a permis de proposer un processus d'adaptation. Nous nous intéressons à deux aspects : quoi adapter ? Et comment réaliser l'adaptation ?

En conséquence, nous définissons la notion de service d'une application et les données échangées entre ce service et l'utilisateur. Lors de l'invocation d'un service, le processus consiste à sélectionner la forme ou l'instanciation adéquate de ce service parmi les formes existantes.

Dans le cas du changement des valeurs des informations contextuelles, le système exécute automatiquement les services concernés, en tenant compte des préférences d'activité de l'utilisateur. L'adaptation permet aussi de modifier les propriétés des données présentées à l'utilisateur selon ses préférences de résultat et d'affichage, et selon les caractéristiques de son dispositif d'accès.

Pour mieux illustrer l'objectif de ce travail, nous utilisons un exemple d'application, conçue et développée pour un PDA ou un téléphone mobile évolué (Smartphone). Nous avons choisi cet exemple car il exprime le besoin d'adaptation au contexte dans les systèmes d'information pervasifs. Nous nous appuyons sur le projet de recherche « Conférence Assistant » de Dey, Salber et Abow **[Dey, Salber et al, 1999]**. Le but de ce projet est de concevoir un organisateur mobile portable pour aider les participants à organiser leur parcours dans les conférences de recherches.

Cette application nous a permis de valider notre modèle de contexte et notre processus d'adaptation fonctionnelle. Nous constatons que dans la plupart des cas, il ne suffit pas seulement de développer une autre version des services pour les terminaux mobiles (selon leurs capacités matérielles et logicielles). En effet, les résultats obtenus dépendent du changement de l'environnement, des souhaits de l'utilisateur ou de la qualité de la communication. La conception et le développement de cet exemple, montre la nécessité et l'intérêt d'avoir une architecture qui assure l'adaptation des applications à de nouveaux contextes d'utilisation.

3. Plan du mémoire

Outre l'introduction générale et la conclusion, notre mémoire est organisé en quatre chapitres.

Le Chapitre 1 présente un bilan sur la notion de contexte dans le domaine de l'informatique ubiquitaire, sur les systèmes d'information pervasifs et sur la sensibilité au contexte (ou context-awareness).

Une étude des approches de modélisation du contexte d'utilisation est aussi élaborée. Nous terminons par une conclusion sur notre étude sur le domaine.

Le Chapitre 2 est consacré à la présentation de l'architecture générale d'un système sensible au contexte ainsi que quelques architectures existantes qui aident au développement de ce genre de systèmes.

Nous faisons aussi un survol de la notion d'adaptation et sa classification et nous terminons par montrer le processus d'adaptation dans les systèmes pervasifs.

Le Chapitre 3 est consacré à la présentation de notre proposition. Nous résumons les travaux similaires qui nous ont inspirés. Notre approche pour la modélisation du contexte qui inclut la gestion des préférences de l'utilisateur est décrite ainsi que le processus de l'adaptation au contexte.

Dans le Chapitre 4, nous présentons tout d'abord l'exemple d'application sensible au contexte que nous avons choisi pour la validation, les caractéristiques du dispositif sensé contenir cette application et les outils utilisés pour le développement.

Modélisation du contexte des systèmes d'information pervasifs

Dans ce chapitre, nous nous intéressons à la modélisation du contexte dans les systèmes pervasifs. Nous commençons par présenter l'informatique pervasive, les différentes définitions de la notion de contexte et de la sensibilité au contexte (ou context-awareness).

Ce chapitre propose aussi, un bilan sur la modélisation du contexte dans le domaine de l'informatique pervasive.

I.1 Exemple illustratif

L'adaptation des logiciels n'est pas un problème nouveau. Mais l'imprévu, qui prévaut dans les systèmes d'information pervasifs (SIP), lui confère un relief particulier. D'où le retour au premier plan d'une notion introduite dans les tous premiers systèmes d'exploitation : le contexte. Pour ces systèmes, le contexte dénotait l'ensemble des informations nécessaires et suffisantes pour traiter une interruption et reprendre l'exécution du programme interrompu. Dans les systèmes d'information pervasifs (SIP), cet ensemble est difficile à cerner puisque le nécessaire et le suffisant sont conditionnés par la finalité visée. Le scénario de la figure 1 illustre l'ampleur de la tâche **[Rey, 2006] :**

Un voyageur pense avoir un peu de temps pour visiter le musée Beaubourg avant de prendre le train du retour. Il s'approche du plan du quartier, énonce (ou saisit) « Beaubourg » au moyen de son assistant personnel. En réponse, le plan montre le chemin optimal, transférable sur le dispositif personnel. Le système indique en privé le temps que notre voyageur peut raisonnablement dédier à sa visite. Dans cet exemple, la connexion de l'assistant personnel avec le plan est réalisée automatiquement, par proximité. La réponse à la demande dépend de la localisation du voyageur (quelque part à Paris). Elle dépend aussi des contraintes notées dans son agenda (ne pas oublier de passer à une pharmacie), et de l'heure de départ du train enregistrée sur le billet électronique.

Cet exemple montre l'importance des informations contextuelles pour une réponse adaptée. Quelles sont-elles ? Intuitivement, sont nécessaires dans ce scénario :

- la détection de la proximité de l'utilisateur avec le plan en sorte que le système pervasif découvre l'arrivée d'un assistant personnel et de là, établisse une connexion avec un service d'aide à la navigation ;
- la connaissance du lieu et de l'heure de la demande, les contraintes et préférences de l'utilisateur afin d'éviter au voyageur de saisir des paramètres ;

- les caractéristiques des ressources d'interaction (plan et assistant personnel) pour que le système ajuste la visualisation du chemin mais aussi juge du caractère privé ou public du dispositif de rendu.

Mais ces informations sont-elles suffisantes ? Seront-elles toujours disponibles en ces mêmes circonstances ? En d'autres circonstances, est-ce-que nous aurons affaire au même ensemble?

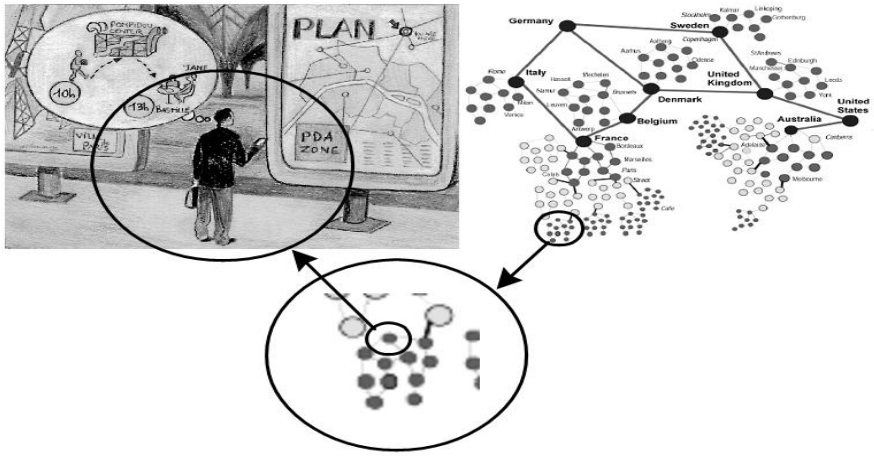


Figure 1. Exemple tiré du scénario du projet européen GLOSS

L'exemple, pourtant simple, de la figure 1, montre la difficulté de l'entreprise. C'est que la notion de contexte est utile pour de nombreux services logiciels et ceci quel que soit leur niveau d'abstraction. Reprenant cet exemple, nous pouvons citer : les services de connexion à l'infrastructure et la découverte de ressources d'interaction, l'authentification et les contrôles d'accès, la protection de l'espace privé, la composition dynamique de services d'information (système d'information géographique et navigation, agenda et préférences personnelles, achat de titres de transport).

I.2. Présentation des systèmes d'information pervasifs

La notion des systèmes d'information pervasifs a été proposée au départ comme une vision future de l'informatique par Mark Weiser **[Weiser, 1991]** suite à son analyse du marché mondial des ordinateurs et des équipements informatiques (Figure 2). Il a remarqué une croissance énorme du nombre d'ordinateurs : un ordinateur pour plusieurs utilisateurs, puis un ordinateur par utilisateur ensuite, plusieurs ordinateurs par utilisateur en utilisant l'intégration des processeurs dans des objets de la vie quotidienne (systèmes embarqués). Cette nouvelle forme de l'informatique qu'il a nommée informatique ubiquitaire (en anglais

“ubiquitous computing”) et dont l’objet viserait à assister implicitement et discrètement un utilisateur dans les tâches qu’il accomplit au quotidien, devenant ainsi la base des systèmes d’information pervasifs **[Cliquet, 2007]**.

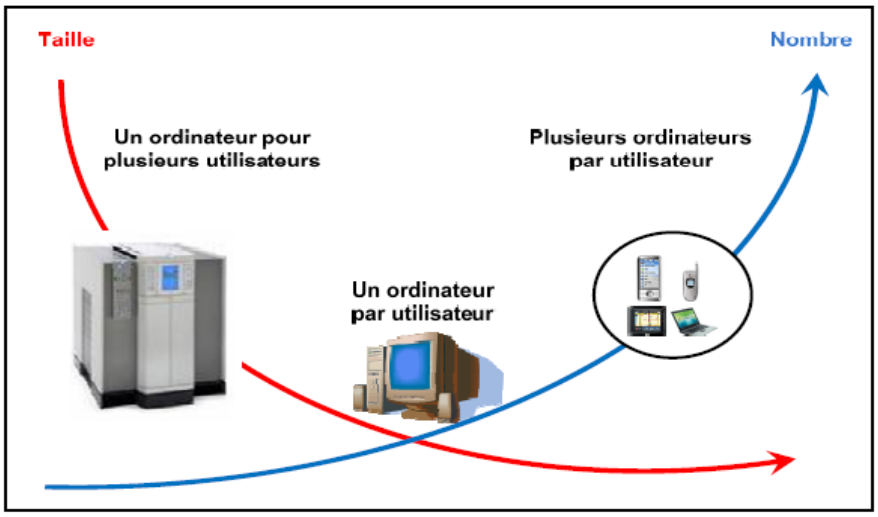


Figure 2. Évolution du marché mondial des ordinateurs. Adaptée de **[Riveill, 2002]**

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

L’avènement des réseaux de communications sans fil avec les standards de télécommunication, tel que GSM, particulièrement utilisés pour la téléphonie mobile, mais également avec l’apparition de WiFi et Bluetooth pour les équipements informatiques et particulièrement pour les terminaux informatiques mobiles tels que les assistants personnels (PDA : personal digital assistant) et les téléphones cellulaires (iPhone), a permis à ces équipements de communiquer (profil matériel, contexte, etc.) pour coopérer. Ceci se fait d’une manière transparente pour l’utilisateur sans son intervention explicite et lui offre la possibilité de se concentrer sur sa tâche principale au lieu de configurer et de gérer l’ensemble des équipements informatiques mis à sa disposition.

L’informatique pervasive favorise par exemple la création d’environnements intelligents tels que la maison intelligente capable de gérer automatiquement les différents équipements présents au domicile de l’utilisateur **[Miraoui, 2009]**.

Les systèmes pervasifs est un domaine particulier de l’informatique résultant de la convergence des travaux existant dans les domaines de l’informatique mobile et des systèmes

distribués. La Figure3, donne un aperçu des apports des systèmes distribués, de l'informatique mobile ainsi que les avancées nécessaires à la réalisation d'un système pervasif.

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

L'informatique mobile est le résultat de l'intégration de la technologie cellulaire et du web pour donner la possibilité aux utilisateurs d'accéder à l'information à n'importe quel endroit et en utilisant différents équipements de petites tailles et de faibles coûts. C'est en quelque sorte la deuxième étape vers l'informatique pervasive « le n'importe où, le n'importe quand » **[Satyanarayanan, 2001]**.

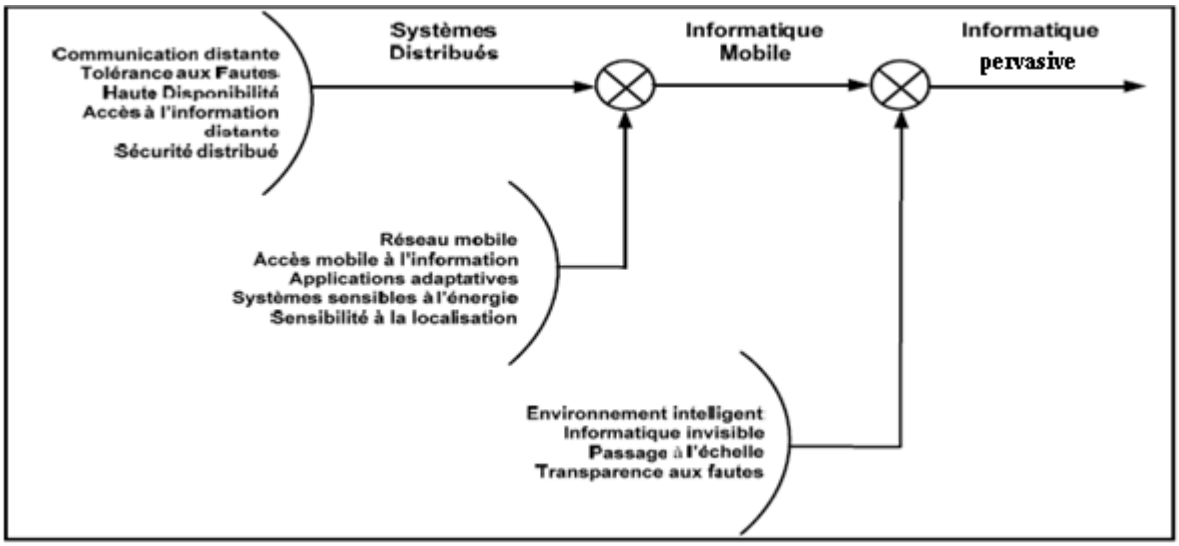


Figure 3. Évolution vers l'informatique pervasive Adaptée de **[Satyanarayanan, 2001]**

Contrairement à l'informatique traditionnelle qui suppose qu'un utilisateur effectue une tâche définie dans un environnement déterminé, l'informatique pervasive repose sur la connaissance du contexte de l'usager pour lui délivrer le service approprié au moment opportun. Ainsi, le badge actif (ActiveBadge) **[Want et al., 1992]**, est l'une des premières applications de l'informatique pervasive : elle permet l'accès par identification à certains bureaux du PARC (Palo Alto Research Center / Centre de recherches de Xerox) et localise un usager afin de lui transférer ses appels téléphoniques dans le bureau où il se trouve. Depuis, les applications basées sur la géolocalisation se sont multipliées, et le guidage routier par GPS (Global Positioning System / Système de guidage par satellites) en est un exemple.

Le contexte ne peut cependant être circonscrit à la seule localisation d'un usager dans l'espace ; d'une manière plus générale, il doit être envisagé **[Dey et Abowd, 2000]** comme " l'ensemble des informations qui peuvent être utilisées pour caractériser la situation courante d'une entité, tel qu'un individu, un lieu ou un objet considéré comme ayant un rapport avec le service délivré par le système".

" MyCampus " **[Sadeh et al., 2005]** exploite ainsi différentes données issues de l'environnement pour définir le contexte courant de l'usager. Ce système en vigueur sur le campus de Carnegie Mellon se compose de plusieurs "agents" qui remplissent des fonctions spécifiques. L'agent "concierge" indique dans quel restaurant universitaire l'étudiant peut prendre ses repas en fonction de ses préférences gastronomiques, de l'heure, de sa localisation sur le campus et des conditions météorologiques. L'agent " réunion " illustre quant à lui, un autre aspect des systèmes pervasifs : celui de l'échange direct de machines à machines ; il assiste les usagers dans le choix d'une date commune de réunion, en parcourant leurs agendas respectifs.

Plus généralement, les applications pervasives relèvent de l'assistance discrète et permanente d'un usager dans l'ensemble de ses activités courantes **[Pascoe, 1998]** ; en ce sens, elles dépassent le cadre actuel de l'informatique en mobilité.

Pour éviter tout risque de confusion entre les termes utilisés dans le domaine de l'informatique pervasive, nous présentons les définitions suivantes qui le caractérisent **[Laforest et Mouël, 2005/2006]**; **[Saha et Mukherjee, 2003]**; **[Weiser, 1993]** :

- **ubiquitaire** : Accessible de n'importe où ;
- **mobile** : Qui intègre les terminaux mobiles ;
- **sensible au contexte** : Qui prend en compte le contexte d'exécution ;
- **pervasif** : Qui associe ubiquité, mobilité et sensibilité au contexte ;
- **ambiante** : Qui est intégré dans les objets quotidiens.

La Figure 4 illustre les composants d'un système pervasif:

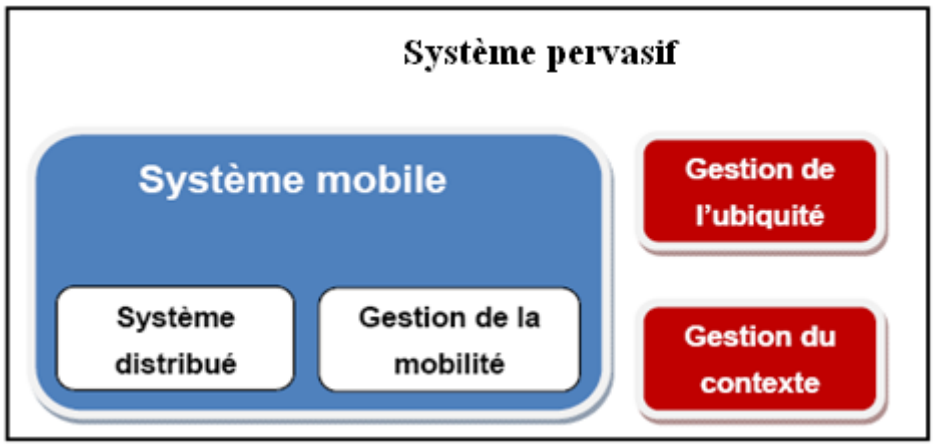


Figure 4. Les composants d'un système pervasif. Adaptée de **[Saha et Mukherjee, 2003]**

L'informatique pervasive met en œuvre cinq éléments de base **[Laforest, 2007] :**

- **Scalabilité:** consiste au passage à l'échelle en gérant des volumes de plus en plus grands d'utilisateurs, d'applications et d'appareils connectés. Permet de développer des applications dont le cœur est indépendant du volume, des utilisateurs et des appareils et d'utiliser des techniques d'adaptation pour pouvoir répondre à chaque cas
- **Invisibilité:** est la nécessité d'un minimum d'intervention humaine pour s'adapter seul aux changements et l'auto-apprentissage. Par exemple : reconfiguration dynamique des caractéristiques réseau d'un appareil, accès aux ressources d'un « espace » en fonction de l'appartenance à la zone géographique de cet espace.
- **Sensibilité au contexte (Context-awareness):** La perception de l'environnement pour interagir plus « naturellement » avec l'utilisateur grâce aux capteurs de l'environnement physique, les matériels auto-descriptifs, la description des personnes et les métadonnées sur les applications.
- **L'intelligence (Smartness) :** Percevoir le contexte d'exécution ne suffit pas, il faut utiliser efficacement les informations du contexte. Exemple : maison intelligente.
- **Pro-action:** Suggérer, proposer des actions correctives à l'utilisateur en fonction du contexte présent ou prédit. Par exemple se déplacer de 100 mètres pour atteindre un réseau plus performant et ainsi accomplir une tâche dans un temps « correct ». Il faut faire attention à balancer avec l'invisibilité! Ceci sous-entend de savoir prévoir un événement, une situation, d'évaluer une situation courante ou possible, comparer deux situations et juger de la meilleure et voir si « ça vaut le coup » de transgresser l'invisibilité.

I.3. Contexte et informatique pervasive

Le contexte, nous l'avons vu en introduction, n'est pas un concept nouveau en informatique : dès les années soixante, systèmes d'exploitation, théorie des langages et intelligence artificielle exploitent déjà cette notion. Avec l'émergence de l'informatique pervasive, le terme est redécouvert et placé au cœur des débats sans pour autant faire l'objet d'une définition consensuelle claire et définitive.

I.3.1. Définitions du contexte

Une étude sur la sensibilité au contexte dans l'informatique mobile réalisée par **[Chen et Kotz, 2000]** a permis de montrer que les définitions existantes du contexte sont des définitions générales, vagues et n'aident pas beaucoup à comprendre ce concept dans un environnement informatique.

Le contexte considéré comme une information sur l'environnement d'un système informatique, ou bien comme conditions qui déterminent un événement nous semble sans limite apparente. En effet, qu'obtiendrons-nous si nous envisageons de décrire dans le moindre détail les composants d'un système informatique ?

De plus, il peut sembler ambitieux de vouloir décrire l'ensemble de conditions qui régissent le déclenchement d'un événement donné. C'est pour ces raisons et bien d'autres que des chercheurs, non satisfaits des définitions générales, ont essayé de définir ce terme pour permettre son utilisation dans leurs recherches. Pour faire suite à notre recherche, nous allons présenter dans l'ordre chronologique l'apparition d'une liste non exhaustive des définitions du contexte dans le domaine de l'informatique. **[Miraoui 2009]**

Schilit et ses co-auteurs **[Schilit, 1994 ; Schilit et al. 1994]** introduisent l'expression context-aware pour désigner un système doté d'un modèle du contexte. Le contexte, selon Schilit, inclut la localisation et l'identité des personnes et des objets à proximité ainsi que les modifications pouvant intervenir sur ces objets. Etudier le contexte, c'est répondre aux questions « Où es-tu ? », « Avec qui es-tu ? », « De quelles ressources disposes-tu à proximité ? ». Il définit donc le contexte comme les changements de l'environnement physique, de l'utilisateur et des ressources de calcul. Ces idées sont reprises par **[Pascoe, 1998]** puis par **[Dey et al., 1999]**.

Un peu plus tard, Brown **[Brown, 1996]** restreint le contexte aux éléments de l'environnement de l'utilisateur, puis il introduit l'heure, la saison, la température, l'identité et la localisation de l'utilisateur **[Brown et al., 1997]**.

Parallèlement aux travaux de Brown, des définitions émergent avec l'introduction explicite du temps et la notion d'état. Ryan et Pascoe **[Ryan et Pascoe, 1997]** assimilent le contexte à l'environnement, l'identité et la localisation de l'utilisateur ainsi que le temps. Ward et ses co-auteurs **[Ward et al., 1997]** voient le contexte comme les états des environnements possibles de l'application.

En 1998, Pascoe définit le contexte comme n sous-ensembles d'états physiques et conceptuels ayant un intérêt pour une entité particulière **[Pascoe, 1998]**.

Puis Dey **[Dey et al., 1999]** précise la nature des entités en question « Le contexte couvre toutes les informations pouvant être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un lieu, ou un objet qui peut être pertinent pour l'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application ».

Cette définition encapsule toutes les autres définitions précédentes puisqu'elle est d'ordre très générique. Dey explique cette généricité du fait que les paramètres du contexte peuvent être implicites ou explicites. En effet, les paramètres du contexte peuvent être fournis directement par l'utilisateur ou par des capteurs situés dans l'environnement de l'utilisateur et de l'application comme ils peuvent parvenir d'une interprétation plus ou moins complexe de ces paramètres. Dey a veillé à ce que sa définition englobe toute donnée implicite ou explicite qui peut être utile à l'application **[Chaari, 2007]**.

Dans leur étude sur la plasticité des IHM, Thevenin et ses co-auteurs **[Thevenin et al., 1999]** aboutissent à une définition assez proche avec la notion de contexte d'interaction. Le contexte d'interaction est un triplet <plate-forme – environnement – utilisateur> où l'environnement est l'ensemble des entités (objets, personnes et événements) périphériques à la tâche courante et pouvant avoir un impact sur le comportement du système ou de l'utilisateur.

En 2001, Winograd **[Winograd, 2001]** approuve la définition donnée par Dey et affirme qu'elle couvre tous les travaux existants sur le contexte. Cependant, il considère que l'utilisation d'expressions comme "toute information" et "caractériser une entité" reste d'ordre très général et ne trace aucune limite à la notion de contexte (tout peut être contexte). Pour apporter plus de précision par rapport à la définition de Dey, Winograd propose : le contexte est un ensemble d'informations. Cet ensemble est structuré et partagé ; il évolue et sert l'interprétation. Il détaille cette définition en disant : "La considération d'une information

comme contexte est due à la manière dont elle est utilisée et non à ses propriétés inhérentes". Il appuie cette idée par l'exemple : "le voltage des lignes d'électricité fait partie du contexte si le système en dépend; sinon, il ne peut être qu'un paramètre quelconque de l'environnement"**[Chaari, 2007].**

Chaari **[Chaari, 2007]** considère que cette définition de Winograd est plus utile pour l'exploitation du contexte dans les applications. Il a repris les principes de cette définition et il a apporté encore plus de précisions pour tracer des limites entre les données propres à l'application et son contexte d'utilisation. Il fournit une nouvelle définition du contexte : " Le contexte est l'ensemble des paramètres externes à l'application qui peuvent influer sur son comportement en définissant de nouvelles vues sur ses données et ses fonctionnalités. Ces paramètres peuvent être dynamiques et peuvent donc changer durant l'utilisation de l'application".

Miraoui et Tadj **[Miraoui, 2009]** font l'abstraction de la notion du contexte et le voient du point de vue des services offerts par un système d'information pervasif. Ils définissent les informations de contexte comme suit : "Toute information dont le changement de sa valeur déclenche un service ou change la qualité (forme) d'un service " **(Miraoui et Tadj, 2007b**).

Miraoui **[Miraoui, 2009]** considère que sa définition est plus abstraite que les autres définitions dans le sens ou elle n'énumère pas des exemples du contexte et limite l'ensemble des informations contextuelles à celles qui sont reliées aux services (objectif d'un système pervasif). Elle ne prend donc pas en compte les autres informations qui peuvent caractériser le contexte, mais ne jouent pas un rôle crucial pour l'adaptation des services.

I.3.2. Contexte pertinent

Un contexte pertinent pour une application est un sous ensemble du contexte observé dont les changements de valeur affectent cette application. En d'autres termes, du point de vue de l'application tous les contextes observables ne sont pas pertinents pour l'application. L'ensemble des contextes observables sélectionnés pour une application donnée du fait que cette dernière est sensible à leurs changements de valeur est appelé contexte pertinent.

I.3.3. Utilisation du contexte

Le contexte est pris en compte dans différent domaines informatiques incluant le traitement du langage naturel, l'apprentissage automatique, la vision par ordinateur, l'extraction de l'information, l'informatique pervasive et même la sécurité informatique **[Miraoui 2009]**.

Tout comme dans l'interaction entre humains, le but de la prise en compte du contexte est de renforcer l'adaptabilité et le support à la décision du système. Chalmers **[Chalmers, 2004]** identifie cinq principales utilisations du contexte dans les systèmes pervasifs comme suit :

1. Senseur du contexte : où le contexte est capté et les informations décrivant le contexte courant (température, localisation, …) sont présentés à l'utilisateur ;
2. Associer le contexte aux données, encore appelé augmentation contextuelle. Par exemple : les enregistrements sur les objets inspectés peuvent être associés à leur localisation ; les notes d'une réunion peuvent être associées aux personnes assistant à la réunion et le lieu où elle s'est déroulée ;
3. Permettre la découverte de ressources contextuelles, par exemple, faire en sorte que l'impression d'un document ait lieu sur l'imprimante la plus proche ;
4. Dans le cas des évènements déclenchés par le contexte : pour déclencher les actions telles que le chargement de données cartographiques à l'entrée dans une région ;
5. Utiliser le contexte pour modifier un service ou pour proposer des services spécialisés. Par exemple pour décrire les limites et les préférences dans une large variété de données offertes, et ainsi pouvoir afficher les données les plus appropriées.

Dans ce mémoire, nous nous intéressons aux applications qui utilisent le contexte pour offrir des services spécialisés et délivrer le contenu qui soit le plus approprié au contexte d'utilisation.

I.3.4. Catégories de contexte

Etant donnée la diversité des informations composant le contexte, il est utile d'essayer de les classer par catégorie pour faciliter leur utilisation. Dans cette section, nous présentons une classification de Chaari **[Chaari, 2007]** qui synthétise les informations contextuelles utilisées dans les solutions existantes. Les entités principales concernées par la notion de contexte sont des lieux, des personnes ou des objets.

Les lieux sont des régions d'espaces géographiques comme des chambres, des bureaux, des bâtiments, des rues ou des zones bien définies. Les personnes peuvent êtres des individus ou des groupes d'individus rassemblés ou répartis. Les objets peuvent êtres des entités physiques, des composants logiciels ou des artefacts (applications, fichiers, ressources…). Chaari classe les informations contextuelles utilisées dans la majorité des travaux existants en quatre catégories principales : identité, localisation, état (ou activité) et temps.

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

La localisation ne se limite pas à la position 2D des objets et des personnes. En effet, elle peut concerner l'orientation, l'altitude et les relations spatiales entre les entités (comme les relations de proximité, de co-existence et de contenance). Par exemple, le fait qu'un objet A est orienté vers un objet B ou un autre objet C est une information qui peut être classée dans la catégorie localisation. La localisation peut aussi référencer des lieux. Ces lieux peuvent être identifiés par leurs emplacements géographiques absolus ou relatifs par rapport à des objets de référence.

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

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

Les données encapsulées par ces quatre catégories peuvent être interprétées ou corrélées pour obtenir des informations contextuelles supplémentaires afin de garantir une évaluation plus étendue d'une situation. Par exemple, en connaissant dans une salle, le nombre de

personnes, leurs positions relatives et l'amplitude du bruit, nous pouvons déterminer si elles sont en conférence ou non.

Les différentes catégories identifiées dans les travaux existants peuvent être utilisées de différentes manières pour assurer la sensibilité au contexte dans une application. Nous détaillons dans le paragraphe suivant cette notion de "sensibilité de contexte".

I.4. Notion de la sensibilité au contexte (Context-Awareness)

La caractéristique principale des systèmes informatiques pervasifs est le changement dynamique de leurs environnements ou bien plus précisément leurs contextes. Pour mieux aider l'utilisateur dans ces tâches quotidiennes, les systèmes informatiques pervasifs doivent tenir compte du contexte global et adapter leurs services aux utilisateurs selon ce dernier. Cette aptitude est connue sous le nom de « sensibilité au contexte » (ou *context-awareness* en anglais) **[Miraoui, 2009]**. Dans la partie suivante, nous allons faire un bref survol des définitions existantes de ce terme.

Le terme sensibilité au contexte a été évoqué pour la première fois par **[Schilit et al., 1994]** dans leurs travaux sur un système de localisation. Ils ont défini la sensibilité au contexte comme l'aptitude d'une application à s'adapter au contexte de son exécution selon : la localisation, l'ensemble des personnes à proximité, les machines, les équipements accessibles, de même que les changements de ces objets dans le temps.

Brown **[Brown, 1996]** a défini la sensibilité au contexte dans son travail relatifs à un guide touristique comme toute application qui prend en compte le contexte de l'utilisateur.

Abowd **[Abowd, 1999]** dit qu'une application sensible au contexte doit percevoir la situation de l'utilisateur dans son environnement et adapter par conséquent son comportement à la situation en question.

Dey **[Dey, 2001]** l'a défini comme un système qui utilise le contexte pour fournir des informations et/ou des services pertinents à l'utilisateur. Selon lui la pertinence dépend de la tâche de l'utilisateur.

Chen et Kotz **[Chen et Kotz, 2000]** ont donné deux définitions. Il y a selon eux :

1. Sensibilité active au contexte : une application qui s'adapte automatiquement au contexte découvert par le changement du comportement de l'application ;

2. Sensibilité passive au contexte : une application qui présente le nouveau contexte ou celui mis-à-jour à un utilisateur intéressé ou rendre le contexte persistant pour une utilisation ultérieure.

Malgré tous les travaux dans le domaine de la sensibilité au contexte, il est loin d'être au point. En effet, plusieurs éléments sont encore à approfondir et à étudier. Winograd **[Winograd, 2001]** a déjà énuméré trois lacunes dans ce domaine : (i) la notion de contexte n'est pas encore bien définie; (ii) les travaux existants manquent de modèles et de méthodes conceptuelles; (iii) il n'y a pas d'outils disponibles pour le développement et l'hébergement des applications sensibles au contexte.

I.5. Modélisation du contexte

Une caractéristique importante des systèmes pervasifs est leur sensibilité au contexte. Celle-ci leur permet d'offrir pro-activement des services adaptés au contexte global (de l'utilisateur, des applications, …). Le contexte doit être bien compris et modélisé sous une forme appropriée pour favoriser son partage entre les différents équipements informatiques du système et offrir un haut niveau d'abstraction pour faciliter la tâche d'adaptation. Cette modélisation consiste à faire l'analyse et la conception de l'information contextuelle contenue dans le système sous forme d'une représentation abstraite au niveau de la structure de données et au niveau de la sémantique. Plusieurs approches de modélisation ayant des particularités liées à la technique utilisée ont été proposées dans la littérature **[Miraoui, 2009]**.

Dans cette section, nous allons présenter, analyser et classifier certaines approches.

I.5.1. Les modèles Attribut/Valeur

La représentation attribut/valeur est la structure des données la plus simple pour la modélisation des informations contextuelles. Elle a été proposée au départ par **[Schilit et al., 1994]** pour la gestion des informations contextuelle d'un environnement **[Miraoui, 2009]**.

Plusieurs architectures présentent le contexte sous forme de paires (attribut, valeur). L'attribut représente un nom d'une information contextuelle. La valeur représente la valeur actuelle de cette information. Par exemple, (Name="*context1*", User="doctorEH102", Localisation="Edouard Herriot Hospital", Time="Mon Jul 09 16:51:20 CEST 2007"). Le contexte *context1* est défini par "l'utilisateur x est localisé dans un emplacement y à un temps t". Cette méthode présente l'avantage de la facilité d'implantation. En effet, la gestion du contexte revient à parcourir la liste des contextes disponibles. Cependant, ce modèle manque d'expression et de complétude.

En effet, sa structure trop « plate » ne permet pas de définir tous les aspects contextuels de l'application. Ce genre de modèle est aussi une source de conflits. Par exemple, si nous définissons un nouveau contexte (Name="*context2*", User="x", Localisation="z", Time="t") avec l'emplacement *z* est « dans l'hôpital Edouard Herriot » (par exemple la chambre 220 de l'hôpital), nous ne pouvons pas dire que *context2* est un sous contexte de *context1* et que toutes les fonctionnalités offertes par l'application dans *context1* doivent aussi exister dans *context2* **[Chaari, 2007]**. Mais cette approche est très utilisée dans les services distribués (les services sont décrits en général avec une liste d'attributs sous forme d'attribut/valeur et la découverte de service est ensuite appliquée par utilisation d'un algorithme qui utilise ces paires attribut/valeur) **[Held et al., 2002].**

I.5.2. Les modèles de représentation par balises

Cette représentation est formée d'une structure de donnée hiérarchique constituée de balises avec des attributs et des contenus. Ces derniers peuvent être définis récursivement par d'autres balises. La profondeur de cette structure dépend du contexte décrit. Comme elle est formée de balises, elle utilise des langages dérivés du SGML (Standard Generic Markup Language) en particulier, le XML (eXtended Markup Language). Pour faire suivre notre idée, nous allons présenter trois langages qui utilisent cette approche.

- **ConteXtML :** Le ConteXtML (Contexte Markup Language) est un protocole simple basé sur XML pour échanger des informations contextuelles entre un client mobile et un serveur. Les messages ConteXtML sont regroupés dans des balises <context> ou des éléments. Ci-dessous un exemple (Figure 5) proposé par Ryan **[Ryan, 2007]** qui présente un message ConteXtML qui peut être envoyé par un client pour indiquer sa localisation courante (l'élément <spatial>). Cette position est décrite à l'aide des coordonnées x, y et z.

```
<context session="123" action="update">
<spatial proj="UTM" zone="33" datum="Euro 1950 (mean)">
<point x="281993" y="4686790" z="205" />>
</spatial>
<require>
<note>
<data name="landuse" value="pasture" />
</note>
</require>
</context>
```

Figure 5. Exemple un message ConteXtML.

CC/PP : Le langage CC/PP (Composite Capabilities / Preferences Profiles), est une recommandation de W3C (World Wide Web Consortium) dans le cadre des travaux sur « device dependance » pour supporter la négociation de contenu entre un navigateur web et un serveur. Il est basé sur RDF (Resource Description Framework) qui est utilisé afin de créer des profils décrivant les capacités des terminaux et les préférences d'un agent utilisateur. CC/PP est utilisé pour personnaliser le contenu sur la base de ses capacités et ses préférences. La figure 6 montre un exemple CC/PP utilisé pour la modélisation de la localisation de l'utilisateur **[Indulska et al., 2003]** :

```
[LocationProfile  [PhysicalLocation [Country, State, City, Suburb]]
[LogicalLocation [IPAddress]]
[GeodeticLocation [Longitude, Latitude, Altitude]]
[Orientation [Heading, Pitch]] ]
```

Figure 6. Modélisation de la localisation de l'utilisateur en utilisant CC/PP

Les extensions du langage CC/PP

Certaines tentatives d'extension de CC/PP ont été effectuées par Held et ses co-auteurs **[Held et al. 2002]** et Indulska et ses co-auteurs **[Indulska et al., 2003]**.

Held et ses co-auteurs ont proposé le *Comprehensive Structured Context Profile* (CSCP) qui permet d'avoir une description du contexte non limitée à deux niveaux hiérarchiques. Cette proposition ne permet pas elle non plus de décrire des relations, des contraintes ou des dépendances entre les informations de contexte.

Les travaux de Indulska et ses co-auteurs ont étendu le vocabulaire de CC/PP pour pouvoir décrire la localisation, les caractéristiques du réseau et les dépendances d'une application. Ainsi, cette modélisation peut être utilisée pour décrire les contextes observables associés à une application sensible au contexte, mais Indulska et ses co-auteurs ont conclu que malgré cette extension, cette modélisation reste non intuitive et difficile à utiliser pour décrire des informations complexes.

I.5.3. Les modèles graphiques

Cette approche consiste à modéliser les informations contextuelles selon un graphe conceptuel.

Bauer **[Bauer, 2003]** a utilisé une représentation graphique basé sur UML (Unified Modeling Language) pour la modélisation des informations contextuelles pour le système de contrôle du trafique aérien.

Indulska et ses co-auteurs **[Indulska et al., 2003]** ont proposé un modèle graphique qui se base sur le formalisme «entité/association ». Les éléments du contexte (utilisateur, machine, réseau, etc.) sont représentés par des entités avec des attributs et en associations entre eux (une association peut aussi avoir des attributs). Ils ont fourni aussi une notation graphique pour leur concept de modélisation.

Henricksen et ses co-auteurs **[Henricksen et Indulska, 2004]**; **[Henricksen et al., 2005]** ont développé une approche de modélisation graphique basée sur la méthode ORM (Object Role Modeling). C'est une méthode orientée « fait » pour l'analyse de l'information au niveau conceptuel, en particulier pour les bases de données relationnelles. La modélisation dans ORM consiste à identifier les types des faits appropriés et les rôles des types d'entités. Selon les auteurs, cette extension est plus formelle et plus expressive pour capter différents types d'informations contextuelles ; elle appuie le raisonnement sur le contexte, décrit bien les informations imparfaites et résout l'ambiguïté dans l'information contextuelle. Cette méthode qui a subi des améliorations est devenue CML (Context Modeling Language) **[Henricksen et Indulska, 2006]** et apparaît comme une extension vers une représentation basée sur XML, XCML **[Robinson et al., 2007]**.

Virgilio et Torlone **[Virgilio et Torlone, 2006]** ont présenté le GPM (General profile Model) pour la description de représentations hétérogènes de données web d'une manière uniforme.

Dans GPM, un profil est une description d'un aspect autonome du contexte dans lequel le site web est visité et il doit influencer la livraison de son contenu. Des exemples de profils peuvent être l'utilisateur, la machine, la localisation, etc. GPM présente un ensemble limité de primitives de base pour la description graphique d'une représentation conceptuelle du contexte.

Les primitives de base sont : profil, dimension, attribut, type de base, choix, séquence ordonnée, séquence non ordonnée et cardinalité. Selon les auteurs, GPM peut être utilisé pour décrire plusieurs contextes d'une manière uniforme et fournit un outil puissant pour la conception et l'analyse des applications sensibles au contexte. De plus, ces auteurs ont défini un processus permettant de traduire les profils d'un modèle à un autre en utilisant GPM comme niveau de représentation intermédiaire.

Sheng et Benatallah **[Sheng et al., 2005]** ont proposé un méta-modèle basé sur une extension d'UML qui permet de modéliser le contexte auquel des services web sont sensibles. Ce langage est appelé ContextUML. Ce méta-modèle est composé de plusieurs classes qui permettent de créer des services sensibles au contexte. La classe Context permet de décrire un contexte observable. Ce dernier peut être un contexte de bas niveau représenté par la classe *AtomicContext*, ou bien un contexte interprété représenté par la classe *CompositeContext*. Pour chaque contexte de bas niveau, le modèle permet de spécifier la source à partir de laquelle il a été collecté.

Le méta-modèle ContextUML permet aussi la description des actions d'adaptation et les situations pertinentes qui permettent de les déclencher. Le méta-modèle ContextUML est caractérisé par sa généricité. Mais, la description des relations de dérivation et de dépendance entre les informations de contexte n'a pas été considérée. De plus, ce méta-modèle n'offre pas le moyen de décrire la qualité des informations de contexte ni leur validité temporelle **[Behlouli, 2007]**.

Malgré la simplicité de ces modèles au niveau de la représentation et des moyens d'expression du contexte, ils restent moins formels que les autres méthodes et n'offre aucune approche empirique.

I.5.4. Les modèles orientés objets

Bouzy et Cazenave **[Bouzy et Cazenave, 1997]** ont proposé un mécanisme orienté objet pour la modélisation du contexte pour simplifier la représentation des connaissances dans des systèmes complexes comme le cas du jeu GO (un jeu très connu depuis 4000 années au Japon, en Chine et en Corée). Ils ont montré que le contexte a plusieurs types et ont donné des exemples : temporel, but, spécial et global. Ils ont justifié leur approche de modélisation orientée objet par sa capacité d'héritage et de réutilisation. Celles-ci permettent de définir un

petit nombre de propriétés, fonctions et règles dans le but de simplifier la représentation des connaissances dans des systèmes complexes.

Cheverest et ses co-auteurs **[Cheverest et al., 2000]** ont proposé dans le cadre du projet GUIDE (un guide touristique sensible au contexte) une approche fondée sur l'intégration d'un modèle orienté objet et un modèle d'information hypertexte. Leur modèle est constitué de deux objets distincts : l'objet point de navigation et l'objet localisation. Ils ont limité l'information significative du contexte à la localisation. Ils ne couvrent pas l'aspect général du contexte.

Le but de la modélisation par l'approche orientée objet est de profiter de la puissance offerte par les mécanismes d'encapsulation, d'héritage…etc. Cette modélisation offre la possibilité de représenter le contexte comme une hiérarchie et de décrire chaque élément indépendamment des autres en utilisant la technique d'encapsulation. Cela permet de favoriser la réutilisation de ces éléments dans d'autres applications mais ne donne pas la possibilité de représenter les relations entre les objets. Cette approche de modélisation est bien efficace en termes de distribution et d'abstraction. Cependant, elle reste propre à une application spécifique et ne permet pas de partager le contexte entre applications.

I.5.5. Les modèles logiques

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

La première approche de modélisation du contexte en utilisant la logique a été publiée en 1994 par McCarthy et son équipe **[McCarthy et Buvac, 1994]**. McCarthy définit le contexte comme une entité mathématique abstraite ayant des propriétés. Cette formalisation logique est fondée sur une réification du contexte et un méta-prédicat *ist* ; *ist(p,c)* signifie que l'assertion *p* est vraie dans le contexte *c*.

Par exemple la formalisation c : *(ist(contextof("Histoire de Sherlock Holmes"),"Sherlock Holmes est un détective")* considère que le personnage Sherlock Holmes est un détective dans l'histoire de Sherlock Holmes. Ce type de modélisation est utilisé dans le domaine de

l'intelligence artificielle où les connaissances sont regroupées en micro-théories **[Guha, 1992]**; selon les valeurs du contexte, on se place dans ou hors d'une micro-théorie **[Behlouli, 2007]**.

Les mécanismes de déductions des solutions basées sur une modélisation logique sont les mécanismes les mieux adaptés pour réaliser l'abstraction des informations en concepts. Cependant, l'inférence sur un ensemble de prédicats, est une opération qui nécessite un grand nombre de ressources systèmes. Cela affecte la consommation d'énergie. Inversement, l'informatique pervasive considère généralement des environnements dans lesquels les terminaux sont limités en ressources (terminaux mobiles). Cette représentation est non structurée. Les solutions qui utilisent ce type de modèle s'appuient souvent sur une gestion du contexte centralisé.

I.5.6. Les modèles basés sur les ontologies

Une ontologie est un ensemble structuré de concepts. Les concepts sont organisés dans un graphe dont les relations peuvent être des relations sémantiques ou des relations de composition et d'héritage.

L'objectif d'une ontologie est de modéliser un ensemble de connaissances dans un domaine donné, c'est-à-dire un choix quant à la manière de décrire un domaine sous une forme utilisable par une machine. Elle fournit un vocabulaire représentatif pour un domaine donné, un ensemble de définitions et d'axiomes qui contraignent le sens des termes de ce vocabulaire de manière suffisante pour permettre une interprétation consistante des données représentées au moyen de ce vocabulaire.

Les ontologies constituent de puissantes ressources pour partager la connaissance. Elles ont généré un grand intérêt en intelligence artificielle et une grande variété de disciplines qui sont confrontés au problème d'intégration d'information et d'interprétation de données.

Une ontologie Formelle est un ensemble de blocs à partir desquels des modèles du monde sont construits. Un agent ou un robot autonome utilisant un modèle particulier ne peut percevoir que la partie du monde que son ontologie est capable de représenter. Seuls les concepts de son ontologie existent pour cet agent. Une ontologie formelle est le niveau de base d'un schéma de représentation des connaissances. Des ontologies formelles ont été construites pour un nombre croissant de disciplines scientifiques, de l'ingénierie ou de la

production. Par analogie aux modèles orientés objets, les ontologies permettent de structurer les concepts et les propriétés de la même manière que les classes et les attributs.

Le langage OWL (Ontology Web Language) est une récente recommandation du W3C. Il permet de décrire des ontologies. Tout comme CC/PP, le langage OWL est basé sur un schéma RDF **[Miraoui, 2009]**. Nous présentons ci-après trois ontologies de contexte : COBRA-ONT, CONON et CoOL.

- CoBrA : Chen et son équipe **[Chen et al., 2004]** ont proposé une approche fondée sur l'idée d'un courtier de contexte (Context Broker Architecture ou CoBrA) ; le contexte est entièrement géré et maintenu par un serveur central (d'une bonne performance) qui s'occupe de maintenir l'état du contexte vis-à-vis d'un ensemble de terminaux et d'agents (localisation, activité, etc.). Il définit une collection d'ontologies appelée COBRA-ONT pour la modélisation de contexte dans un environnement d'une salle de rencontre intelligente. COBRA-ONT est exprimée dans le langage OWL qui définit des concepts typiques associés avec des places, des agents et des événements.

Les ontologies jouent un rôle important dans CoBrA. Elles aident le courtier de contexte à partager les informations contextuelles avec d'autres agents et lui permet de raisonner sur le contexte. Le développement de COBRA-ONT se focalise sur la création d'ontologies convenables pour la construction de systèmes sensibles au contexte pragmatique. Dans le but d'appuyer le raisonnement sur les ontologies dans CoBrA, les auteurs ont prototypé un moteur d'inférence OWL appelé F-OWL. Ce moteur d'inférence est implémenté en utilisant Flora-2 qui est un langage de base de connaissance orientée objet et une plateforme de développement des applications qui traduit un langage unifié de F-logic, Hilog, et « transaction logic » au moteur de déduction XSB.

- CONON : Wang et ses co-auteurs **[Wang et al., 2004]** ont proposé une ontologie du contexte CONON (CONtext ONtology) codée en OWL pour la modélisation du contexte dans un environnement pervasif et pour le support de raisonnement logique sur le contexte. CONON fournit une première ontologie du contexte supérieur qui capte les concepts généraux du contexte de base, et fournit aussi l'extensibilité pour ajouter des ontologies spécifique à un domaine particulier d'une manière hiérarchique.

Les auteurs ont jugé que la localisation, l'utilisateur, l'activité et les entités informatiques forment le contexte fondamental pour capturer les informations sur la situation d'exécution.

La deuxième ontologie est une spécialisation de la première pour un domaine du contexte. Elle constitue un ensemble de sous-classes qui viennent se greffer à l'ontologie de base afin de détailler la modélisation suivant un environnement tel que la maison ou le lieu de travail. Alors que l'ontologie de premier niveau définit les termes qui caractérisent l'environnement d'une manière abstraite, la partie de l'ontologie spécifique au domaine est constituée d'un ensemble de termes concrets (TV, PDA, maison, bureau, cuisine, etc.).

L'avantage des ontologies réside dans la possibilité d'effectuer un raisonnement logique grâce à l'apport de nombreux outils. Dans l'approche CONON, le moteur de raisonnement est sollicité pour expliciter la description du contexte (par exemple déterminer de manière explicite qu'un utilisateur se trouve dans une salle, implique que cette même personne se situe également dans le bâtiment où se trouve cette salle !). En plus des règles de déductions de bases utilisées pour les ontologies, l'approche CONON permet d'ajouter des règles personnalisées. Ces règles sont écrites grâce à des prédicats associés à la logique de premier ordre. Elles sont introduites pour permettre de caractériser la situation de l'utilisateur.

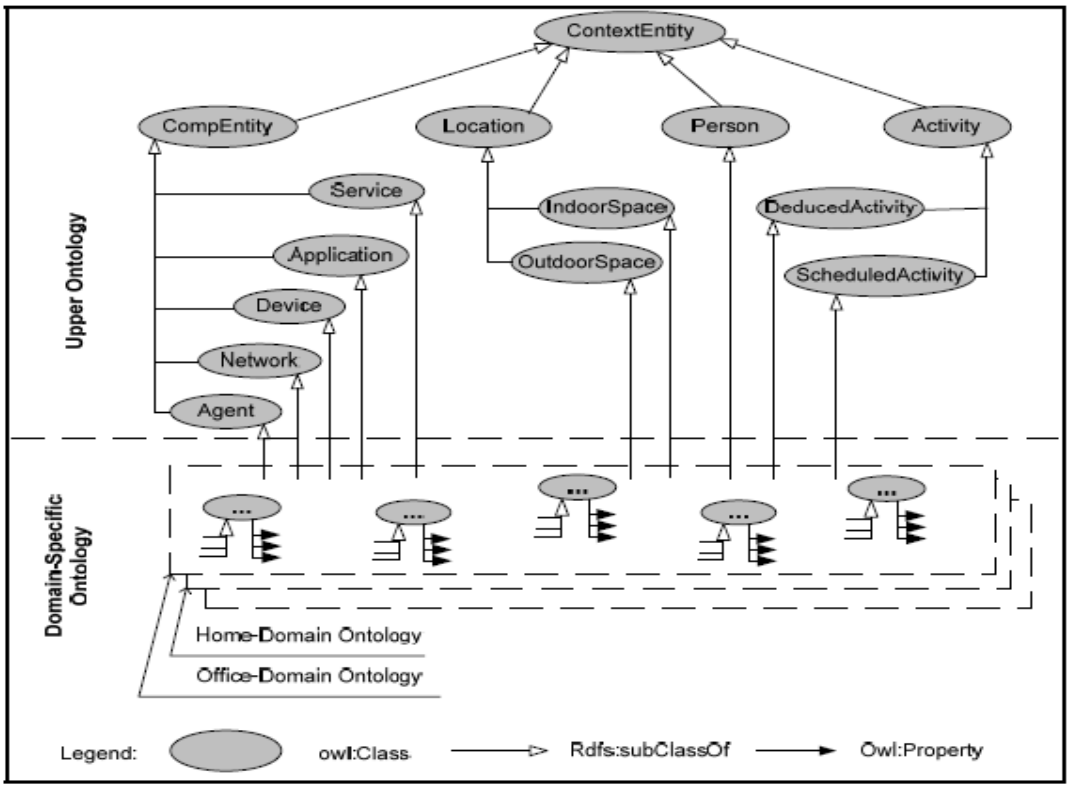


Figure 7. L'ontologie à deux niveaux de l'approche CONON. Adaptée de **[Wang et al., 2004]**

CoOL : Strang et son équipe **[Strang et al., 2003]** ont introduit l'approche CoOL (Context Ontology Language). Celle-ci utilise les ontologies et le modèle de contexte ASC (Aspect-Scale- Context). CoOL est dérivé d'un modèle qui peut être utilisé pour permettre la sensibilité au contexte, l'interopérabilité du contexte durant la découverte des services et l'exécution dans un système à architecture distribuée. L'élément principal de cette architecture et le raisonneur qui infère des conclusions sur le contexte basé sur une ontologie construite avec CoOL. Les auteurs n'ont pas défini CoOL comme un seul langage monolithique, mais comme une collection de plusieurs fragments regroupés dans deux sous ensembles :

- OWL et DAML+OIL sont deux langages basés sur XML et RDF/S. Ils font partie des ontologies du web sémantique.
- F-Logic est un langage logique combinant les caractéristiques de l'orienté objet et de la logique de prédicats.

L'utilisation de plusieurs langages d'ontologies permet du point de vue de représentation des connaissances à un développeur d'utiliser un de ces langages qui lui semble adéquat.

L'avantage des approches qui utilisent une ontologie réside dans les caractéristiques même de l'ontologie, qui offrent non seulement le moyen de faire des descriptions sémantiques, mais aussi de publier les données décrites à travers le réseau. Chacune des approches décrites ci-dessus se focalise sur la création d'un méta-modèle pour classifier les informations de contexte. Cette classification est liée aux applications qui utilisent ces ontologies.

Conclusion

La notion de contexte est extensible à volonté. En pratique, elle n'englobe qu'un nombre limité et variable de caractéristiques. Il s'agit des caractéristiques du dispositif d'accès, ou bien de la localisation de l'utilisateur, ou encore de l'identité de l'utilisateur et de ses préférences. D'autres éléments tels que les bruits environnants, la connexion réseau, la situation de l'utilisateur, etc. influent. Il est important aussi de préciser la finalité visée par la définition du contexte.

Nous avons vu dans ce chapitre que l'un des problèmes cruciaux des systèmes sensibles au contexte est celui de la représentation (modélisation) du contexte. Cette modélisation est souvent dépendante de l'application et il y a peu de travaux « génériques » dans ce domaine.

Nous allons voir, dans le chapitre suivant, la notion d'adaptation et nous allons présenter quelques architectures existantes qui aident au développement et au déploiement des systèmes sensibles au contexte.

Chapitre II

Architectures des systèmes sensibles au contexte

Dans ce chapitre, nous présentons l'architecture générale d'un système sensible au contexte, tirée d'une synthèse des travaux existants. Ensuite, nous analysons quelques

architectures existantes qui aident au développement et au déploiement de ce genre de systèmes. Nous faisons aussi un survol de la notion d'adaptation et sa classification dans la littérature.

Après l'étude des caractéristiques souhaitées des applications sensibles au contexte et des caractéristiques fondamentales des systèmes pervasifs, nous terminons par montrer l'adaptation de contenu et de service dans les systèmes pervasifs.

II.1. Architecture d'un système sensible au contexte

Le développement d'une application sensible au contexte consiste à programmer toutes les étapes de gestion du contexte en commençant par la description des contextes qu'il faut observer, les contextes pertinents auxquels l'application est sensible et les situations pertinentes qui nécessitent l'adaptation de l'application.

La collecte des informations de contexte nécessite de programmer l'interaction de l'application avec différents types de capteurs, alors que l'interprétation et l'analyse du contexte nécessitent de programmer un mécanisme qui étudie constamment les informations observées pour interpréter de nouvelles informations de contexte et détecter leurs variations pertinentes. De plus, un mécanisme d'adaptation doit être programmé pour permettre l'adaptation de l'application dès la détection des situations pertinentes auxquelles l'application est sensible.

Le développement des applications sensibles au contexte, est une tâche complexe qu'il est nécessaire de simplifier. En effet, trois catégories d'architectures existent :

- **Accès direct de l'application aux données brutes des capteurs**. Le câblage entre l'application et les capteurs est implémenté sans couche intermédiaire. Ces systèmes sont difficilement extensibles.
- **Architecture middleware**. Une architecture en couche ou l'application n'a pas accès aux détails de bas niveau des capteurs. Elle facilite l'extension des systèmes et leur réutilisabilité.
- **Architecture client-serveur**. Un serveur de contexte regroupe toutes les données issues des capteurs. Plusieurs clients peuvent requêter le contexte.

Dans ce qui suit, nous détaillons l'architecture générale et les fonctions qu'un système doit accomplir pour qu'il puisse être sensible au contexte.

La conception des applications sensibles au contexte soulève de nouveaux défis **[Chaari, 2007]** : (i) les méthodes classiques de développement des logiciels sont difficiles à appliquer

sur les applications sensibles au contexte; (ii) les solutions proposées pour ce genre d'applications sont soit spécifiques à un besoin précis soit elles manquent d'abstraction lors de leur conception; (iii) la capture du contexte est souvent distribuée et mène à des conceptions complexes.

Dey **[Dey, 1999]** a été le premier à proposer une séparation entre l'acquisition de contexte et son utilisation dans les applications. Il s'appuie sur une étude de nouvelles méthodes de conception des systèmes interactifs mobiles où il y a une séparation entre l'application et le nouveau monde d'interfaces, d'icones, de menus, de pointeurs et de moyens d'interaction sur les terminaux mobiles.

Dans les applications sensibles au contexte, nous somme confronté à des problèmes similaires à ceux rencontrés lors du développement des systèmes interactifs mobiles. Pour cette raison, Dey pense qu'il est utile de séparer la gestion du contexte de l'application, afin de pouvoir développer une plateforme générique de développement et déploiement d'applications sensibles au contexte. La seule chose qui différencie, selon Dey, une application sensible au contexte d'un système interactif mobile est que ce dernier manipule des données explicites qui sont soit des variables internes de l'application soit des entrées explicites de l'utilisateur.

Dey a proposé une plateforme semblable aux systèmes interactifs tout en généralisant les types d'entrées pour prendre en compte des données implicites déduites de l'environnement de l'utilisateur (données du contexte). Inspiré par des concepts utilisés dans des outils d'interaction, Dey a défini quelques abstractions **[Dey, 1999]** qui aident à inférer une interprétation de haut niveau du contexte, et qui supporte la séparation entre la gestion du contexte et l'application.

Cette idée a été reprise par la majorité des travaux actuels **[Matthias, 2007]** dans le domaine de la sensibilité au contexte, proposant ainsi des architectures à plusieurs couches. Ces architectures diffèrent dans les fonctions, les noms et l'emplacement de leurs couches. Cependant, nous remarquons que toutes ces propositions se basent sur cinq couches principales : capture du contexte, interprétation/agrégation du contexte, stockage/historique du contexte, dissémination du contexte et application (figure 1). Dans ce qui suit, nous détaillons les fonctions de ces cinq couches et leurs différents éléments.

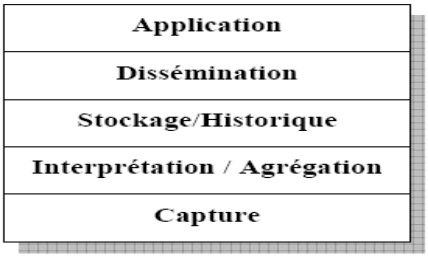


Figure 1 - Architecture générale d'un système sensible au contexte. Adaptée de **[chaari, 2007]**

II.1.1. Couche de capture du contexte

La première couche d'une architecture sensible au contexte est composée d'une collection de capteurs. Un capteur est une source matérielle ou logicielle qui peut générer une information contextuelle. Nous distinguons trois types de capteurs : physiques, virtuels et logiques **[Indulska et al., 2003]**.

1 Capteurs physiques : Les capteurs physiques sont des dispositifs matériels qui sont capables de fournir des données de contexte.

2 Capteurs virtuels : Les capteurs virtuels fournissent des informations contextuelles à partir d'applications ou services logiciels. Par exemple, il est possible de détecter l'emplacement d'un livreur de marchandise en consultant son carnet électronique de rendez-vous sans avoir recours à des capteurs physiques. Nous pouvons aussi détecter l'activité de l'utilisateur sur un microordinateur en analysant les évènements de la souris ou les saisies à partir du clavier. Les capteurs virtuels sont beaucoup moins couteux que les capteurs physiques puisqu'ils sont basés sur des composants logiciels qui sont généralement moins chers que des appareils électroniques.

3 Capteurs logiques : Ce type de capteurs utilise généralement plusieurs sources d'information contextuelles pour fournir une autre information de synthèse plus précise. Ces capteurs peuvent réutiliser des capteurs physiques et virtuels pour fournir un contexte de plus haut niveau. Par exemple, un capteur logique peut fournir l'emplacement d'un caissier dans un grand magasin en analysant les différentes sessions ouvertes sur les caisses du magasin. Dans certain travaux, les capteurs logiques sont considérés comme des interpréteurs de contexte.

Chaque type de capteur doit être attaché à un composant logiciel permettant l'accès aux informations capturées. Ces composants sont généralement fournis avec des pilotes logiciels

(drivers) et une API de communication avec les capteurs. Des interfaces génériques de communication permettent de garantir un accès transparent aux informations capturées **[Bauer, 1998]**. Par exemple, une interface getPosition(), qui retourne la position d'un capteur en mouvement peut être implémentée en utilisant un système GPS ou un système de localisation GSM.

II.1.2. Couche d'Interprétation et d'agrégation du contexte

Cette couche offre des moyens d'interprétation des données contextuelles fournies par les capteurs du contexte. Elle sert à l'analyse et à la transformation des données brutes fournies par la couche de capture du contexte dans d'autres formats de haut niveau qui sont plus faciles à manipuler et à utiliser. En effet, les capteurs fournissent généralement des données techniques qui ne sont pas appropriées pour une utilisation directe par l'application. Les transformations effectuées sur les données brutes fournies par les capteurs peuvent être réalisées par plusieurs opérations : extraction, quantification, raisonnement, agrégation... Par exemple, les coordonnées GPS d'une personne peuvent être moins significatives qu'une adresse physique sous forme de numéro de rue et de ville.

La complexité des interprétations de contexte peut varier d'une simple agrégation de valeurs qui proviennent de plusieurs capteurs à des raisonnements ou analyses statistiques complexes. Par exemple, la localisation de plusieurs personnes dans une seule salle peut inférer le fait qu'ils sont en réunion. Dans ce cas, le niveau de bruit peut aussi être une information importante pour savoir s'ils sont en réunion de travail ou de loisir.

Cette couche doit aussi assurer la résolution de conflits causés par l'utilisation de plusieurs sources de contexte. En effet, ces sources peuvent donner des résultats contradictoires ou peuvent aboutir à des situations imprécises. Cette couche doit donc avoir une certaine intelligence d'interprétation pour résoudre ces conflits **[Dey, 1999]**, **[Kiciman, 2000]**.

II.1.3. Couche de stockage et historique du contexte

La troisième couche "stockage et historique du contexte" organise les données capturées et interprétées et les stocke pour une utilisation ultérieure. Ce stockage peut être centralisé ou distribué. La solution centralisée est l'option la plus répandue et la plus utilisée puisqu'elle facilite la gestion des mises à jours et des variations des valeurs du contexte. La gestion distribuée du contexte est beaucoup plus complexe puisqu'elle inflige des fonctions additionnelles de découvertes de ressources et d'actualisation des valeurs du contexte. De

plus, cette gestion distribuée alourdit la tâche de l'application qui doit gérer la collecte des différentes informations contextuelles d'une façon interne.

Pour stocker une information, nous avons besoin de définir un modèle pour la décrire. Ainsi, un modèle de contexte est requis pour pouvoir l'utiliser dans l'application. Nous avons présenté les approches de modélisation du contexte dans la section 4 du chapitre 1.

II.1.4. Couche dissémination du contexte

Cette couche assure la transmission des différentes informations contextuelles à l'application. Ces informations sont distribuées sur différents lieux géographiques et proviennent de plusieurs types de dispositifs. Ceci a poussé les chercheurs à dédier une couche à la transmission des données à l'application. Cette couche assure une transparence totale de la communication avec l'application. En conséquence, le développement de l'application devient plus simple.

Sans cette couche, on serait amené à développer des protocoles de communications avec les différentes sources de contexte. La couche de dissémination du contexte offre des moyens de communication standards pour notifier l'application des changements de contextes et leur transmission à l'application. Plusieurs systèmes offrent des mécanismes de gestion d'évènements qui se basent essentiellement sur les fonctions de gestion de requêtes directes ou de notification **[Bauer, 1998]**. L'application peut demander un accès direct à une information contextuelle précise (pull) mais elle peut aussi s'abonner pour recevoir tous les changements des valeurs de cette information (publish/subscribe). Ces deux fonctions principales assurent un moyen de communication transparent et efficace pour la dissémination des valeurs de contexte à l'application **[Rodden, 1998]**. L'implantation de ces types de communication est nécessaire pour garantir la sensibilité au contexte dans une application.

II.1.5. Couche application

La couche application dans les systèmes sensibles au contexte existants **[Chen, 2003]** est représentée par l'application qui offre ses services aux différents clients concernés. Cette couche est responsable de l'extraction des informations des différentes sources de données attachées à l'application. Elle doit aussi implémenter les réactions nécessaires aux changements du contexte.

Chaque application s'abonne à la couche de dissémination du contexte pour accéder aux différentes informations contextuelles et être informée de leurs changements. Une application peut accéder à ces informations de deux façons différentes : synchrone et asynchrone.

Dans le cas d'un accès synchrone, l'application demande à la couche de dissémination de lui fournir une information contextuelle précise. Ceci est réalisé généralement par des appels directs de fonctions au niveau de la couche dissémination.

Dans le cas de la communication asynchrone, l'application s'abonne à des évènements spécifiques qui correspondent à des changements de valeurs de contexte. Dès qu'un événement est déclenché, l'application est simplement notifiée ou bien l'un de ses services est directement invoqué en utilisant des fonctions de callback implémentées dans la couche dissémination.

Nous présentons, dans la section suivante de ce chapitre, l'étude de quelques architectures dans le domaine de la sensibilité au contexte.

II.2. Quelques architectures des systèmes sensibles au contexte

L'informatique pervasive constitue un nouveau domaine de l'informatique qui depuis son avènement, exigeait la mise en place de nouveaux outils et de nouvelles mesures (architectures, cadres de travail, intergiciels). Jusqu'alors, ces outils ne faisaient pas partie de l'informatique traditionnelle afin de supporter les particularités de tels systèmes, à savoir : la mobilité, la sensibilité et la variabilité du contexte. L'architecture d'un logiciel décrit l'organisation globale d'un système en termes de composants et de leurs interactions. Elle aide à gérer et à contrôler la complexité du développement de logiciels **[Albin, 2003]**.

Dans la littérature de l'informatique, nous trouvons plusieurs études (« surveys ») des architectures des systèmes sensibles au contexte. Dans le prochain paragraphe, nous allons présenter une étude proposée par Miraoui **[Miraoui, 2009]** qui décrit brièvement les architectures d'un ensemble de systèmes qui ont marqué ; selon lui, l'évolution des systèmes sensibles au contexte.

II.2.1. ActiveBadge (1992)

Le projet ActiveBadge **[Want et al., 1992]** développé par la firme Olivetti a permis la réalisation d'une application permettant l'acheminement des appels téléphoniques à une personne selon sa localisation au téléphone le plus près de lui.

Le système utilise des badges émettant des signaux infrarouges (à une fréquence bien déterminée). Ces badges sont portés par le personnel d'une entreprise et chaque badge contient l'identification du porteur. Les signaux émis par ces badges sont interceptés par des capteurs distribués dans les locaux de l'entreprise. Les signaux perçus par les capteurs sont ensuite envoyés à un serveur. Ce dernier présente à un agent (réceptionniste) les informations de localisation des personnes porteuses de badges ainsi que les locaux où elles se trouvent. Celui-ci achemine les appels reçus pour une personne au téléphone qui se trouve dans le local le plus près de la personne appelée (cette tâche peut être automatisée). L'ActiveBadge est basé sur une architecture distribuée des capteurs (Figure 2) et sur une architecture en quatre couches au niveau du serveur ; ces quatre couches sont les suivantes :

• contrôle du réseau qui surveille le fonctionnement du réseau de capteurs ;

• présentation des données pour le stockage et le contrôle des informations de localisation ;

• traitement des données pour la sélection des informations intéressante lors du changement de localisation ;

• interface utilisateur pour l'affichage des informations textuelles sur le changement de position des badges.

ActiveBadge est plutôt une infrastructure matérielle pour des systèmes sensibles à la localisation.

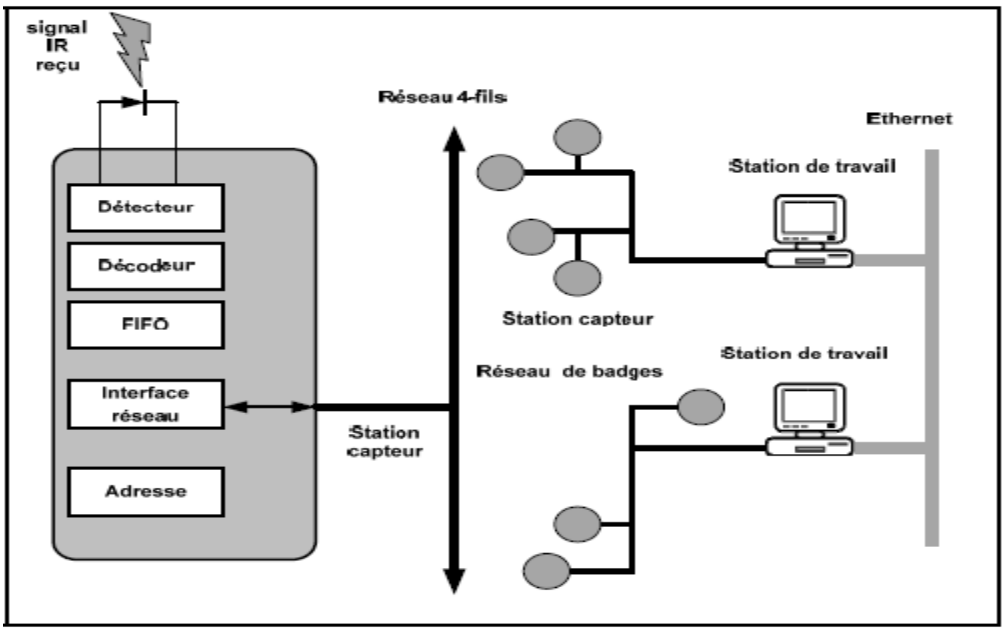


Figure 2. Architecture de l'ActiveBadge. Adaptée de **[Want et al., 1992]**

II.2.2. ParcTab (1995)

Le projet ParcTab de Xerox **[Want et al., 1995]** est une infrastructure qui favorise le développement des applications sensibles au contexte de localisation (localisation d'une personne, matériel qui l'entoure, personnes avoisinants, etc.).

Le ParcTab est un assistant personnel digital (PDA) porté par l'usager et fonctionne comme un terminal graphique. Il est en communication infrarouge avec des émetteurs-récepteurs d'un édifice local. Ces émetteurs-récepteurs communiquent à leur tour (en infrarouge) avec des passerelles (« gateway »). Ces passerelles sont connectées à un réseau local de postes de travail au moyen d'une connexion RS-232. À chaque ParcTab correspond un agent logiciel qui contrôle la communication avec des applications qui s'exécutent sur les postes de travail du réseau local. Cela décharge les ParcTabs des opérations de traitement qui épuisent leurs ressources limitées (Figure 3).

Le système de communication est basé sur le mécanisme d'appel de procédure à distance (RPC) entre les ParcTabs et les applications sur les postes de travail du réseau local. Cette infrastructure favorise le développement d'applications sensibles au contexte, en particulier la localisation telle que la notification des courriels pour un utilisateur selon sa localisation et les personnes avoisinants par un texte qui s'affiche par le ParcTab ou une simple notification par un signal sonore (le cas échéant de filtrer les courriels urgents lorsque l'utilisateur se trouve en conférence ou réunion).

Ses auteurs envisagent aussi plusieurs autres applications sensibles au contexte se basant sur cette infrastructure telles qu'un contrôleur de programme à distance, une collaboration assistée, un accès aux informations et aux ressources selon le contexte et la possibilité d'autres applications. Le ParcTab aussi est un système primitif sensible à la localisation fondé également sur une architecture matérielle.

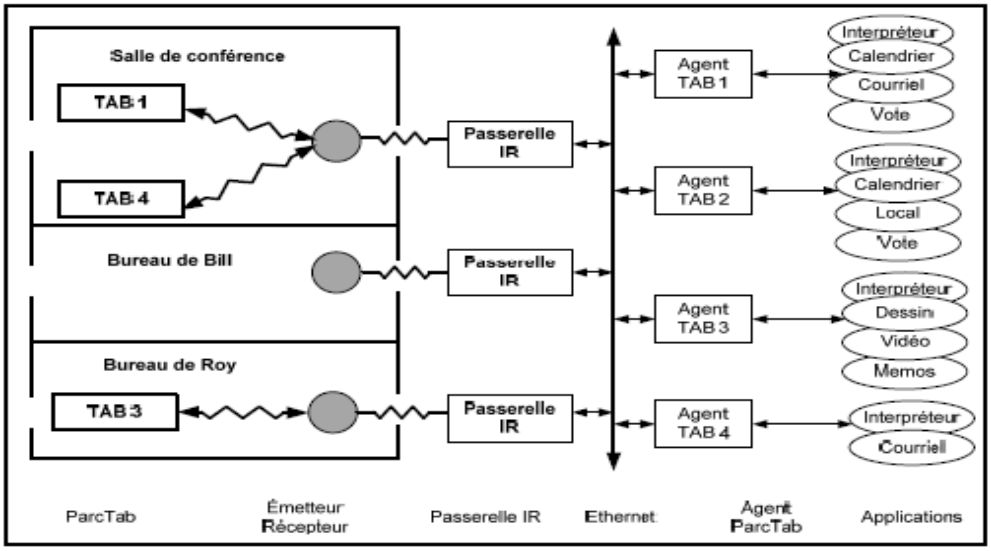


Figure 3. Architecture du ParcTab. Adaptée de **[Want et al., 1995]**

II.2.3. Stick-e-notes (1997)

Le projet stick-e-notes **[Pascoe, 1997]** consiste en un cadre de travail (en anglais : Framework) pour le développement des applications sensibles au contexte principalement de localisation. Il est basé sur l'utilisation d'un ensemble de PDAs connectés à un capteur de localisation (GPS ou ActiveBadge). Ces PDAs peuvent être en communication ou non selon l'application. L'idée de base est inspirée des bouts de papiers collant (stick notes) qu'on met généralement sur les portes, tables, équipements, etc. pour se rappeler de quelque chose ou attirer l'attention de quelqu'un. Ces notes seront électroniques et écrites par l'utilisateur et attachées à des contextes bien précis (exemple : localisation) et sauvegardés dans son PDA. Ces notes sont déclenchées (affichées ou signalées) plus tard lorsque le même contexte apparaît de nouveau (exemple : l'utilisateur attache une note descriptive d'un musée lorsqu'il se trouve dans ce dernier. Chaque fois que l'utilisateur entre dans cette zone géographique, la note descriptive du musée est affichée). Les notes peuvent être de différents formats : texte, pages HTML, son, vidéo, programme à exécuter, etc. Le contenu d'une note est décrit en SGML pour faciliter l'échange et l'extensibilité de l'application. .

Les auteurs ont défini quatre composantes logicielles pour construire le support de stick-enote. Les composants sont les suivants :

- **seprepare** : permettant à l'utilisateur de préparer les notes ;
- **semanage** : offrant la possibilité de la gestion des notes ;

• **setrigger** : donnant la possibilité de déclencher des notes si leurs contextes sont satisfaisants ;

• **seshow** : permet de stocker des notes déclenchées et de les présenter à l'utilisateur.

II.2.4. Cyberguide (1997)

L'idée derrière le projet cyberguide **[Abowd et al., 1997]** est d'équiper l'utilisateur d'un guide touristique électronique sensible à son contexte (localisation, orientation). L'infrastructure matérielle est composée d'un ensemble d'assistants personnels digitaux (PDAs) connectés à des GPS pour détecter la position du touriste. Ces assistants personnels peuvent communiquer entre eux ou avec un réseau local par infrarouge. L'objectif est de guider un touriste dans sa visite touristique en lui offrant les sites intéressants à visiter selon la localisation et les chemins à suivre ainsi que des informations utiles selon sa position. Les composants de l'architecture logicielle sont les suivants :

1. Carte géographique de l'environnement physique que le touriste est en train de visiter avec des représentations des objets intéressants (édifices, tours, etc.) ;

2. Information (librairies) contenant l'ensemble des informations et descriptions des sites que le touriste peut visiter et ayant une symbolisation spéciale sur la carte géographique ;

3. Navigateur permettant de détecter la localisation actuelle du touriste dont le but et de lui offrir des informations sur l'environnement qui l'entoure ;

4. Messager permettant de fournir un service de livraison de messages pour le touriste. Il lui est également permis d'envoyer une question, de recevoir des messages pervasifés ou des suggestions. Il peut aussi entrer en communication avec d'autres touristes.

Un autre projet GUIDE à été proposé par **[Cheverest et al., 2000]** ayant le même objectif que le cyberguide. Les deux projets cyberguide et GUIDE sont deux systèmes spécifiques aux applications de localisation.

II.2.5. Context Toolkit (2001)

Le Context Toolkit **[Dey et al., 2001]** a été proposé pour appuyer le développement des systèmes sensibles au contexte. Il a une architecture en couche pour permettre la séparation de processus d'acquisition et de représentation de contexte de l'adaptation. Il est basé sur les gadgets du contexte (context widgets). Ces gadgets fonctionnent de la même manière que ceux d'une interface utilisateur graphique (GUI) pour cacher la complexité des capteurs physiques utilisés par les applications. Cela donne au contexte plus d'abstraction et fournit

des blocs réutilisables et personnalisés pour la capture du contexte. Les composants de l'architecture (Figure 4) sont :

• **les capteurs** : capture de contexte physique ;

• **les gadgets** : encapsulent les informations contextuelles et fournissent des méthodes pour accéder à ces informations de la même manière que pour les gadgets graphiques ;

• **les interpréteurs** : transforment l'information du contexte dans le but de lever le niveau d'abstraction;

• **le groupeur** : groupement des informations du contexte relatif à un sujet ou à une situation ;

• **le découvreur** : maintient un registre des capacités existantes dans le cadre de travail (les composantes actuellement disponibles pour utilisation par des applications) ;

• **le service** : exécute des actions au profit des applications.

Cette architecture est simple à implémenter. Elle offre une communication distribuée entre les équipements du système et des gadgets réutilisables.

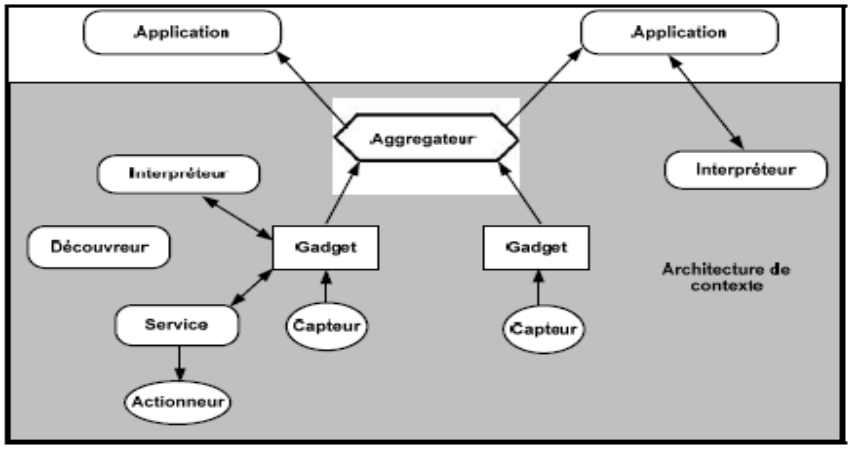


Figure 4. Éléments de l'architecture du Contexte Toolkit. Adaptée de **[Dey et al., 2001]**

II.2.6. Context management framework (2003)

Le CMF (context management framework) **[Korpipää et al., 2003]** permet de faire le raisonnement sur le contexte sémantique en temps réel en présence d'incertitude de bruit et le changement rapide des informations ; délivrer le contexte aux applications selon un mécanisme de communication basé sur les événements. Le CMF propose une architecture client/serveur comme modèle de communication entre les entités qui le composent.

L'architecture (Figure 5) contient cinq composants de base comme suit **[Miraoui, 2009]** :

1. **Le Gestionnaire du contexte** : stockage des informations contextuelles (serveur) et livraison aux clients selon différents mécanismes (requête/réponse, inscription/notification etc.) ;

2. **Le Serveur de ressources** : acquisition des informations de contexte de capteurs physiques et leur interprétation selon un schéma spécifique avant de les envoyer au gestionnaire de contexte ;

3. **Le Service de reconnaissance du contexte** : conversion d'un flux de données à une représentation définie dans l'ontologie du contexte ;

4. **Le Service de détection des changements** : détecter les changements d'un service suite à un changement de contexte ;

5. **La Sécurité** : vérifie et contrôle les informations de contexte.

Le Context Management Framework se base sur un gestionnaire de contexte centralisé qui communique avec tous les autres modules de la plateforme. En effet, dans CMF, le gestionnaire de contexte récupère les informations contextuelles à l'aide du serveur de ressources. Ensuite, il les interprète en utilisant le service de reconnaissance du contexte. Enfin, il les diffuse à l'application.

L'application peut communiquer avec le gestionnaire de contexte de trois manières différentes:

- elle envoie des requêtes directes au gestionnaire de contexte. Dans ce cas, le gestionnaire de contexte délègue cette requête au serveur de ressources.

- elle s'abonne aux changements des valeurs de contexte. Dans ce cas, le gestionnaire de contexte utilise le service de détection des changements de contexte.

- elle demande un contexte de haut niveau (composé) en utilisant le service de reconnaissance du contexte.

Les auteurs de CMF proposent d'utiliser la logique floue pour obtenir des informations de haut niveau et ainsi enrichir le contexte et la description de l'environnement de l'utilisateur **[Chaari, 2007]**. Le CMF utilise les ontologies pour la représentation du contexte, mais n'offre pas un mécanisme raisonnement sur le contexte.

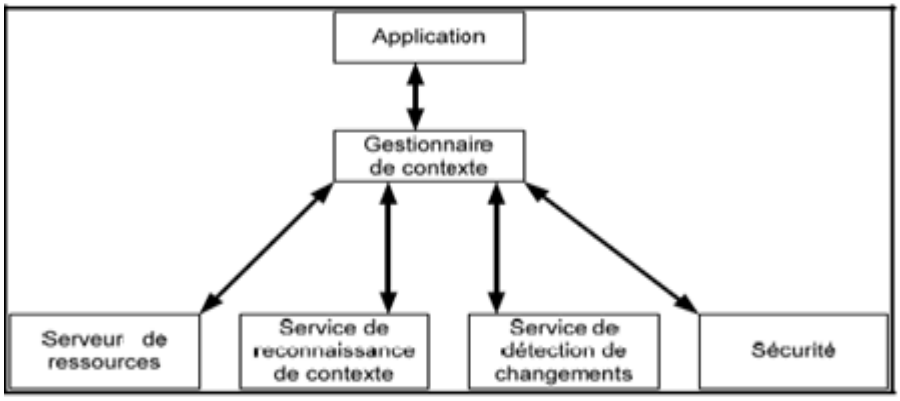


Figure 5. Architecture de CMF. Adaptée de **[Korpipää et al., 2003]**

II.2.7. Hydrogen (2003)

Hydrogen **[Hofer et al., 2003]** est une architecture et un framework développés pour appuyer la sensibilité au contexte. C'est une architecture en trois couches pour convenir aux besoins particuliers des équipements mobiles composés des couches (Figure 6) : adaptateur, gestion (serveur de contexte) et application. Le serveur de contexte contient toutes les informations captées par la couche adaptateur. Il fournit à la couche application, ou à d'autres équipements, le contexte requis en utilisant le modèle de communication égale-à-égale.

L'approche Hydrogen considère le contexte comme toute information pertinente sur l'environnement d'une application et décrit le contexte par un modèle orienté objet.

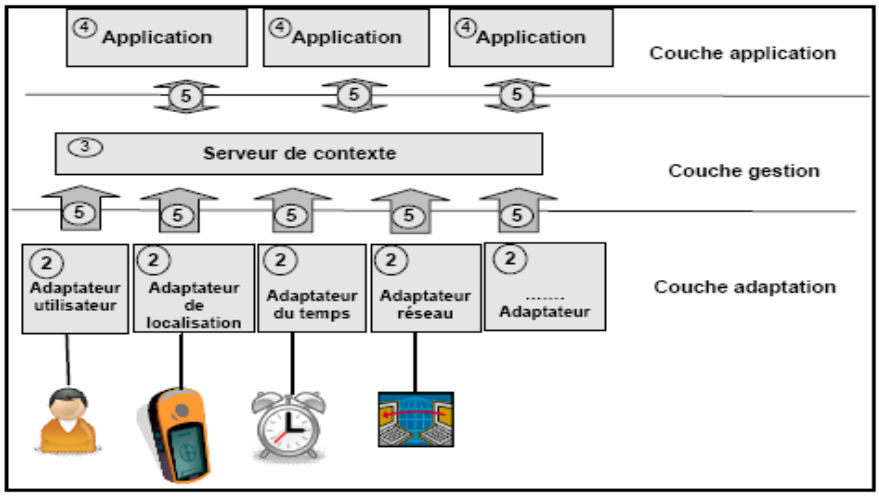


Figure 6. Architecture de Hydrogen. Adaptée de **[Hofer et al., 2003]**

II.2.8 CASS (2004)

L'outil CASS (Context-awareness sub-structure) **[Fahy et Clarke, 2004]** est un intergiciel (middleware) pour appuyer le développement des applications sensibles au contexte. Il offre une bonne abstraction des informations contextuelles et utilise un modèle orienté objet pour la représentation du contexte.

Cette architecture (Figure 7) est basée sur un serveur contenant une base de données des informations contextuelles ainsi qu'une base de connaissances. Celle-ci permet au moteur d'inférence d'inférer d'autres informations de contexte en utilisant la technique de chaînage arrière. Les équipements mobiles sont munis de différents types de capteurs. Ceux-ci perçoivent les changements du contexte et les envoient au serveur sans aucun traitement local.

La communication entre les équipements mobiles et le serveur est assurée par des connexions sans fils. Le serveur contient aussi un module pour l'interprétation du contexte perçu. Cette architecture offre une bonne modularité permettant à un utilisateur de modifier facilement les composants du serveur, en particulier le mécanisme d'inférence.

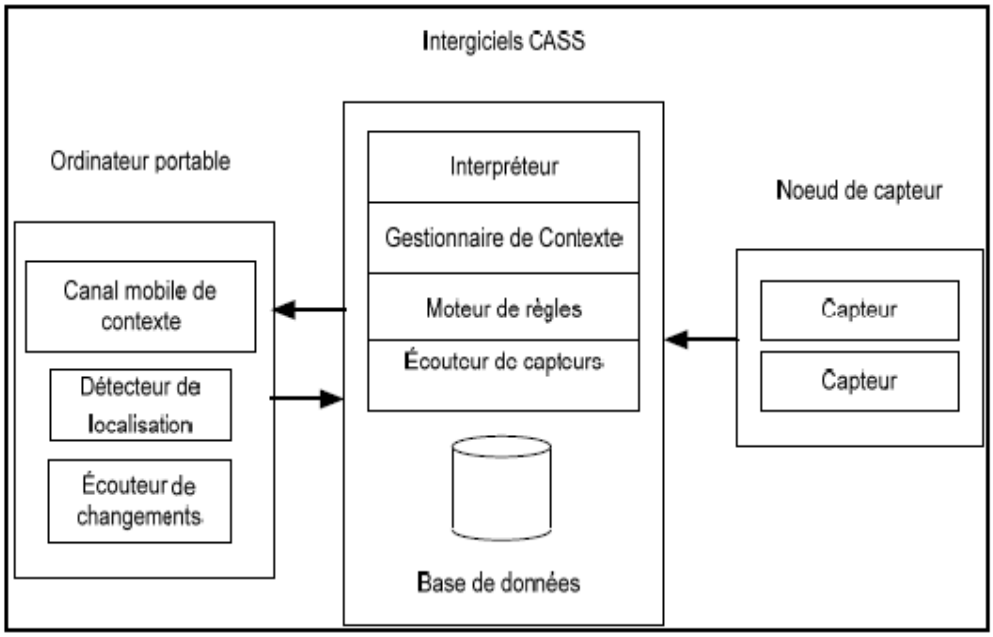


Figure 7. Architecture de CASS. Adaptée de **[Fahy et Clarke, 2004]**

II.2.9. CORTEX (2004)

Biegel et Cahill **[Biegel et Cahill, 2004]** ont proposé un Framework pour faciliter le développement des applications mobiles sensibles au contexte dit CORTEX (COoperating Real-time sentTient objects: architecture and EXperimental evaluation). L'architecture est basée sur les «sentient objects ». Elles ont des caractéristiques avantageuses pour un environnement informatique pervasif : (i) Elles sont douées de sensation en ayant la capacité de percevoir l'état de l'environnement par des capteurs ; (ii) Elles sont autonomes en ayant la capacité de fonctionner indépendamment du contrôle humain d'une manière décentralisé ; (iii) Elles sont proactives en prenant des initiatives pour accomplir un but proposé.

L'architecture d'un « sentient object » (Figure 8) est composée de deux interfaces : capture des événements perçus par les capteurs, émission des événements pour s'adapter au contexte actuel. Cette architecture contient un module pour la fusion du contexte et son interprétation dans un haut niveau d'abstraction. Elle contient également, un module pour la représentation hiérarchique du contexte dont le but est de limiter le contexte de la situation actuelle et par la suite de limiter les actions possibles. Un moteur d'inférence pour spécifier le comportement de l'application à un contexte donné qui utilise le modèle d'exécution « événement-condition-action ».

La communication entre les différents «sentient object », capteurs et actuateurs qui composent le système utilise le mécanisme basé sur l'événement qui s'établit d'une manière dynamique pendant le fonctionnement du système. Le système d'inférence est écrit en CLIPS. Cette architecture est une solution ad hoc pour un réseau mobile.

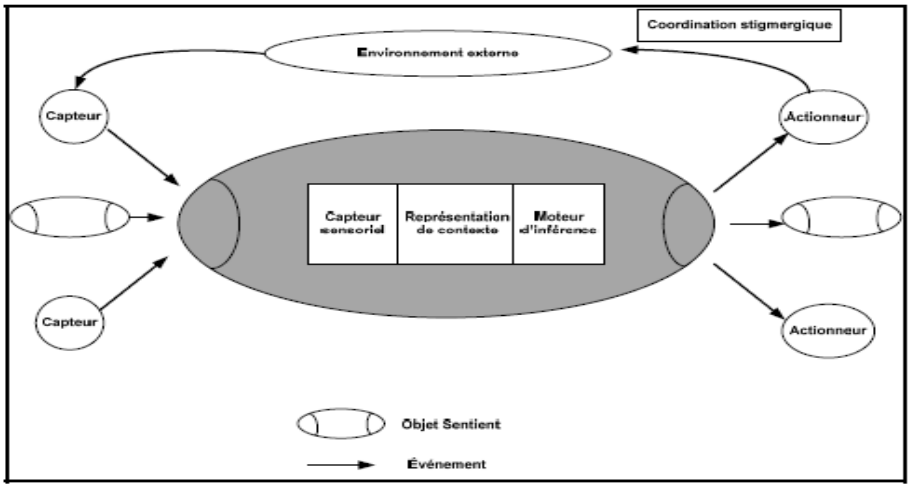


Figure 8. Architecture d'un « sentient object ». Adaptée de **[Biegel et Cahill, 2004]**

II.2.10. SOCAM (2004)

SOCAM (A Service-Oriented Context-Aware Middleware) **[Gu et al., 2004]** est une architecture d'un intergiciel (middleware) sensible au contexte orienté service pour la construction et le prototypage rapide des services mobiles et sensibles au contexte dans un environnement intelligent d'une automobile. L'architecture comprend les composantes suivantes (Figure 9) :

- Fournisseur de contexte : Ils capturent des informations contextuelles utiles de sources hétérogènes de l'environnement de l'utilisateur et de l'application. Ensuite, ils les convertissent en des représentations OWL pour que le contexte puisse être partagé et réutilisé par d'autres composants de l'architecture.

- Interpréteur de contexte : Il fournit des services de raisonnement logique sur les représentations OWL du contexte en appliquant des enchaînements de règles d'interprétation. Il fournit aussi un service d'interrogation intelligent qui permet de résoudre les conflits d'interprétation du contexte.

- Base de données de contexte : Elle stocke les différents éléments de l'ontologie "pervasive" décrivant l'environnement de l'application et les instances des ontologies spécifiques au domaine de l'application décrivant l'environnement de l'utilisateur.

- Services sensibles au contexte : Ces services utilisent les différentes informations stockées dans la base de données de contexte pour modifier leur comportement selon le contexte courant.

- Service de localisation de services : ce service fournit un mécanisme avec lequel des utilisateurs ou des applications peuvent localiser les fournisseurs et les interpréteurs de contexte.

L'architecture utilise le modèle client/serveur. L'interpréteur du contexte collecte les informations du contexte des fournisseurs de contexte (internes et/ou externes) et de la base de données du contexte. Il les livre ensuite au service mobile sensible au contexte et au service de localisation de services. La force principale de l'architecture de SOCAM est son système de raisonnement robuste sur le contexte. Celui-ci utilise les ontologies pour la description du contexte. L'architecture utilise deux classes d'ontologies : spécifiques au domaine et généralisées. Plusieurs systèmes de raisonnement peuvent être incorporés dans l'interprétateur du contexte pour appuyer des tâches variées de raisonnement.

SOCAM utilise l'approche la plus sophistiquée de collecte et de dissémination du contexte : (i) les capteurs sont encapsulés par des services web; (ii) la communication avec l'interpréteur du contexte est réalisée par un échange d'événements en utilisant des représentations OWL des informations échangées. L'architecture est utilisée pour le développement d'une petite application (voiture intelligente).

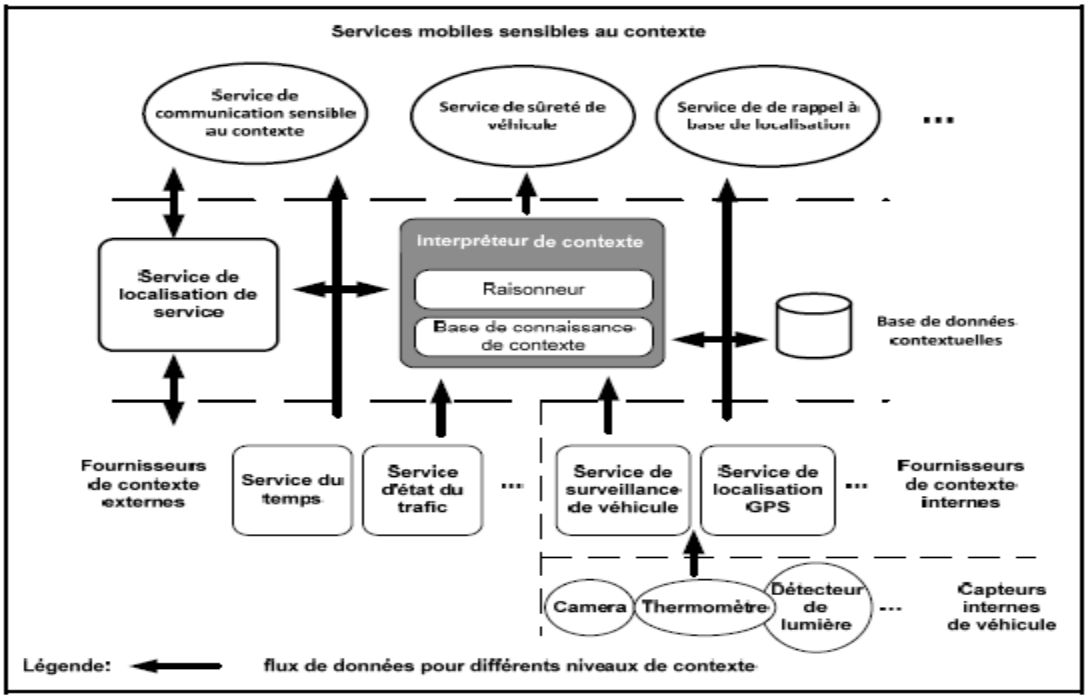


Figure 9. Architecture de SOCAM. Adaptée de **[Gu et al., 2004]**

II.2.11. CoBrA (2004)

CoBrA (Context Broker Architecture) **[Chen, 2004]** est une architecture basée sur un agent courtier (broker) pour appuyer le développement des applications sensibles au contexte dans un espace intelligent. Le courtier est un agent autonome qui gère et contrôle le modèle de contexte d'un domaine spécifique. Il est implanté sur une machine dédiée (serveur) et possède des ressources puissantes.

L'agent courtier possède une architecture en couche comprenant les éléments suivants (Figure 10): base de connaissance du contexte, moteur de raisonnement sur le contexte, module d'acquisition du contexte et module de gestion de confidentialité.

L'agent courtier fait l'acquisition du contexte des équipements, agents (applications) et les capteurs de son environnement. Il les fusionne par la suite dans un modèle cohérent qui sera ensuite partagé par les équipements et leurs agents.

CoBrA utilise les ontologies pour la description du contexte, ce qui permet un raisonnement robuste et un partage efficace des informations du contexte. Elle utilise un modèle centralisé pour le stockage et le traitement du contexte puisque les équipements mobiles dans un système pervasif possèdent des ressources limitées et une politique de confidentialité pour l'utilisateur. Pour cela, l'architecture nécessite un serveur dédié pour l'agent courtier.

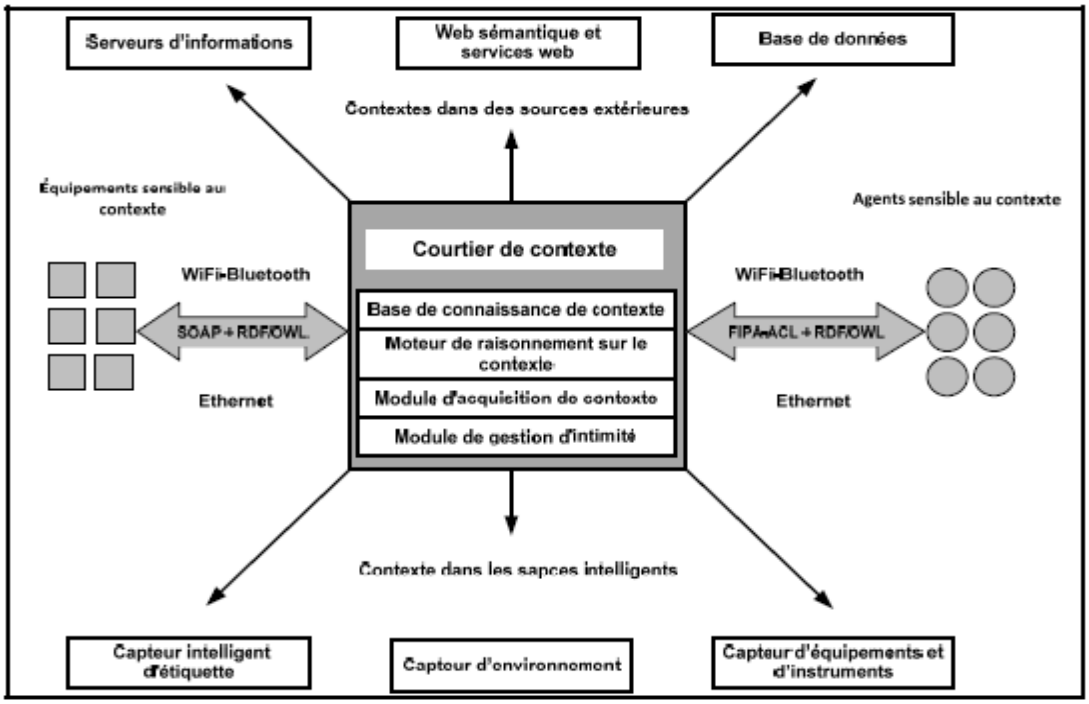


Figure 10. Architecture de CoBrA. Adaptée de **[Chen, 2004]**

II.2.12. JCAF (2005)

[Bardram, 2005] a proposé un cadre de travail (framework) JCAF (Java context awareness framework) basé sur le langage Java dont le but est d'aider et d'appuyer les développeurs des applications sensibles au contexte. L'architecture de JCAF (Figure 11) est composée d'un ensemble de ContextService » communicant selon le modèle d'égale-à-égale (peer-to-peer) et responsable de la collecte du contexte dans un environnement spécifique (chambre, hôpital, laboratoire, etc.).

Un «contextService » est composé de quatre composants :

1. **Le Récipient d'entité** : responsable d'échange d'informations de contexte avec les clients de contexte en utilisant un modèle de communication basé sur l'événement (inscription/notification). Il Contient en plus une ou plusieurs entités décrivant le contexte d'un objet de l'environnement (personne, ordinateur, médecin, … etc.) ;

2. **Le Transformateur** : effectue principalement deux opérations : a) l'agrégation des informations de contexte et b) la traduction entre les types du contexte ;

3. **Les Entités d'environnement** : permettent la communication entre les entités et gèrent l'accès aux ressources partageables ;

4. **Le Contrôleur d'accès** : contrôle l'accès au « ContextService » par une authentification correcte des requêtes de clients pour accéder aux contextes des entités.

Les clients du contexte peuvent être de trois types :

1. **L'écouteur d'entité** : qui peut être une entité d'un autre « ContextService » et qui peut accéder au contexte des entités d'un « ContextService », soit par le schéma requête/réponse, soit par le mécanisme d'événement inscription/notification. Il est possible d'utiliser le mécanisme d'inscription selon le type du contexte ;

2. **Le moniteur du contexte** : permet l'acquisition du contexte par des capteurs et assure une transformation du contexte brut ;

3. **L'actionneur du contexte** : offre la possibilité de commander des actuateurs de l'environnement physique.

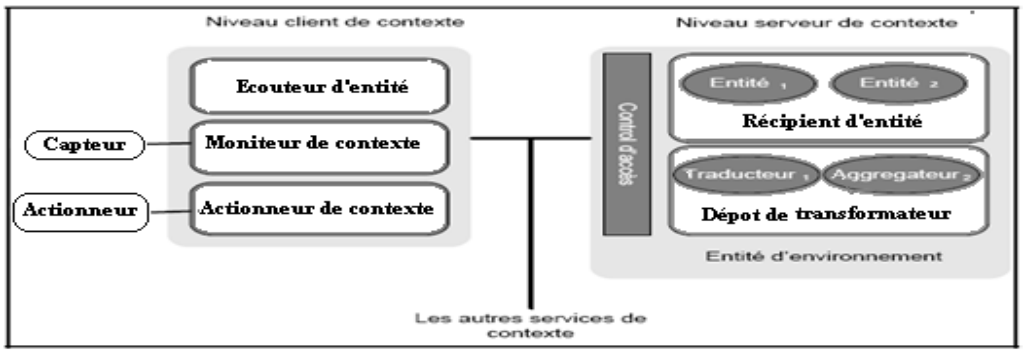


Figure 11. Architecture de JCAF. Adaptée de **[Bardram, 2005]**

Le JCAF assure un contrôle des informations contextuelles (degré de confiance d'une information provenant d'un capteur, probabilité d'erreur d'une information d'un capteur particulier, etc.). La communication à distance entre les composantes de l'architecture est assurée par le mécanisme RMI (remote method invocation) de Java.

Les « ContextService » ne possèdent pas un mécanisme de découverte automatique mais un fichier de configuration qui contient tous les autres «ContextService» actifs. Le JCAF ne contient pas une couche pour raisonner sur le contexte et pour déduire des informations à partir du contexte courant. Il offre cependant des modules réutilisables et portables grâce à l'utilisation de Java.

II.2.13. Analyse des architectures antérieures

Selon Miraoui **[Miraoui, 2009],** la plupart des architectures proposées font une séparation du processus de capture de contexte et son utilisation. Cela permet une abstraction des détails de capture de bas niveau et un accroissement de l'extensibilité et de la réutilisation des systèmes. Parmi les architectures proposées, il y a deux approches selon que les informations contextuelles soient centralisées ou distribuées. La plupart de ces architectures sont en couches et contiennent principalement les éléments suivants :

1. **La capture** : capture physique du contexte par utilisation de différent type de capteurs ;
2. **L'interprétation** : transformation des informations brutes capturées en d'autres plus significatives et plus utilisables ;
3. **Le raisonnement** : (qui n'est pas présent dans toute l'architecture) déduction et prédiction des nouvelles informations du contexte ;
4. **La gestion et le stockage** : opérations de base pour la gestion des informations contextuelles (ajout, suppression, recherche, mise à jour, etc.) ;
5. **L'adaptation** : adaptation de services offerts selon le contexte.

Les architectures proposées sont presque toutes spécifiques à un domaine d'application (systèmes de localisation, interaction personne-machine, etc.) et demandent des efforts supplémentaires pour l'adaptation **[Miraoui, 2009]**. Les architectures fondées sur un serveur de contexte souffrent du problème principal d'un système centralisé : lorsque le serveur tombe en panne, tous les autres composants seront affectés. Rares sont les architectures qui contiennent tous les composants requis cités ci-dessus ou ne reposent pas sur un modèle de contexte solide et fiable pour permettre un raisonnement efficace sur le contexte.

Les architectures existantes ne détaillent pas la couche d'adaptation et insistent sur le fait que la réalisation de cette couche dépend du domaine de l'application et que les services implémentés dépendent fortement de la nature et du besoin des utilisateurs de l'application. Cependant, aucune des architectures proposées ne présente une stratégie complète précisant comment l'application peut s'adapter aux différentes variations du contexte. L'adaptation de l'application au contexte capturé n'est pas une priorité dans ces études. Avant de présenter les types d'adaptation existants, nous présentons les caractéristiques souhaitées des applications sensibles au contexte

II.3. Les caractéristiques souhaitées des applications sensibles au contexte

Selon Lavirotte et son équipe **[Lavirotte et al., 2009],** pour que les utilisateurs adoptent les applications sensibles au contexte:

- L'utilisateur est intéressé par un contexte immédiat.
- Il est nécessaire qu'il y ai un bénéfice et que l'utilisateur puisse garder le contrôle et la capacité de faire un autre choix.
- Concernant la sécurité et le respect de la vie privée, les utilisateurs doivent avoir le contrôle sur les informations qui les concernent. La manière dont les informations sont utilisées doit être clairement identifiée et les utilisateurs peuvent seulement récupérer les informations.

Rhodes et Maes soulignent que la pertinence peut se corréler avec l'utilité mais pas obligatoirement : « La récupération d'un document pertinent mais déjà bien connu n'est pas utile » **[Lavirotte et al., 2009]**.

II.4. Les caractéristiques fondamentales des systèmes pervasifs

D'après l'exemple du chapitre 1, nous pouvons facilement dégager les caractéristiques fondamentales d'un système d'information pervasif. Il offre :

- Un service transparent sur un réseau, en particulier sur internet pour différentes tâches : naviguer ou expédier un courriel (mais dans le cas général ce n'est pas nécessaire que ce soit internet) ;
- Des équipements connectés par différents moyens (WiFi, Bluetooth, …) ;
- Un système qui surveille le contexte et les paramètres du service qu'il offre aux utilisateurs.

Le système doit s'adapter aux différents contextes pour aider l'utilisateur dans ses tâches et lui permettre de se concentrer sur ses tâches principales sans se préoccuper des moyens pour les accomplir ; c'est le cas dans notre exemple :

1. Pour offrir un meilleur service au touriste, le système détecte le type du dispositif d'accès et offre le résultat de la recherche du chemin optimal selon un affichage approprié. La qualité de la communication ici entre le plan et le dispositif est un élément important pour le choix de service (expédition de la carte du chemin ou affichage publique) ;

2. Le type de service dépend dans cet exemple de la localisation de l'utilisateur (détection de la proximité de l'utilisateur pour déclencher le service et utilisation de l'information « lieu de la demande » pour effectuer la recherche) ;

3. La communication entre les équipements est aussi présente dans notre exemple. À l'arrivée du touriste, le système détecte son assistant personnel et effectue une connexion avec ce dernier pour recevoir la requête de l'utilisateur. Une fois la demande reçue, le système cherche d'autres sources de contexte (les contraintes notées dans son agenda et l'heure de départ du train enregistrée sur le billet électronique).

4. La stratégie générale de découverte d'activité utilisée par les systèmes informatiques pervasifs consiste à la lecture d'une base de données (le cas des informations enregistrées dans l'agenda interne de l'assistant personnel et l'heure de départ du train inscrit dans le billet électronique).

II.5. L'adaptation

Les travaux existants dans le domaine de la sensibilité au contexte se sont focalisés sur tous les aspects de capture, interprétation, modélisation, stockage et dissémination du contexte. Cependant, il y a beaucoup moins de travaux qui s'intéressent à l'adaptation et l'actualisation de l'application au contexte. Ce travail se focalise spécialement sur cet aspect d'adaptation au contexte.

Dans cette partie, nous présentons la notion d'adaptation. Nous évoquerons aussi son utilité, ses objectifs et les domaines où elle est utilisée.

II.5.1. Définition de l'adaptation

L'adaptation est, selon le grand dictionnaire, l'opération qui consiste à apporter des modifications à un logiciel ou à un système informatique dans le but d'assurer ses fonctions et, si possible, d'améliorer ses performances, dans un environnement d'utilisation.

Dans le domaine de la sensibilité au contexte, l'adaptation est étendue par la notion d'adaptabilité qui caractérise un système qui est capable de changer son comportement lui-même afin d'améliorer ses performances ou de continuer son exécution dans des environnements différents **[Chaari, 2007]**.

Après avoir mis en œuvre une application, plusieurs raisons peuvent conduire à l'adapter. Ces raisons peuvent être classées en quatre catégories **[Ketfi et al., 2002]** :

1. Adaptation correctionnelle

Dans certains cas, nous remarquons que l'application ne se comporte pas correctement ou comme prévu. La solution est d'identifier le module de l'application qui cause le problème et de le remplacer par une nouvelle version supposée correcte. Cette nouvelle version fournit la même fonctionnalité que l'ancienne, elle se contente simplement de corriger ses défauts.

2. Adaptation adaptative

Même si l'application développée s'exécute correctement, parfois son environnement d'exécution (comme le système d'exploitation), les composants matériels (ou d'autres applications ou données dont elle dépend) changent. Dans ce cas, l'application est adaptée en réponse aux changements affectant son environnement d'exécution.

3. Adaptation évolutive

Au moment du développement de l'application, certaines fonctionnalités ne sont pas prises en compte. Avec l'évolution des besoins de l'utilisateur, l'application doit être étendue avec de nouvelles fonctionnalités. Cette extension peut être réalisée en ajoutant un ou plusieurs modules pour assurer les nouvelles fonctionnalités ou même en modifiant les modules existants pour étendre leurs fonctionnalités tout en gardant l'architecture de l'application.

4. Adaptation perfective

L'objectif de ce type d'adaptation est d'améliorer les performances de l'application. A titre d'exemple, nous nous rendons compte que l'implémentation d'un module n'est pas optimisée. Nous décidons alors de remplacer l'implémentation du module en question. Un autre exemple peut être un module qui reçoit beaucoup de requêtes et qui n'arrive pas à les satisfaire. Pour éviter la dégradation des performances du système, nous diminuons la charge de ce module en installant un autre module qui partage la charge avec le premier.

II.5.2. Classification de l'adaptation

Après avoir présenté les différentes raisons d'utilisation de l'adaptation dans une application, nous énumérons les différentes classifications utilisées dans les travaux existants d'adaptation. Cette classification élaborée par **[Chassot et al., 2006]** dépend de la manière d'application de l'adaptation : statique ou dynamique, centralisée ou distribuée, comportementale ou architecturale **[Chaari, 2007]**.

1. Adaptation statique

L'adaptation statique ou manuelle consiste à préparer plusieurs versions de la même ressource avant son exploitation. Cette ressource peut être un document, une image, une vidéo ou même une application. La ressource est adaptée manuellement avant de pouvoir la réutiliser dans son nouveau contexte.

Cette technique a été adoptée pour plusieurs développements au début de l'apparition des terminaux mobiles afin de pouvoir exploiter des ressources initialement prévues pour des PC standards. Avec une adaptation statique, le résultat est dédié au nouveau contexte d'utilisation de la ressource ce qui garantit la sûreté de son exploitation. Elle présente aussi une solution simple et efficace car chaque fournisseur s'occupe de l'adaptation de la ressource pour ses contextes. Cependant, elle présente l'inconvénient de la difficulté d'évolution pour la prise en charge de nouveaux contextes d'utilisation.

2. Adaptation dynamique

Contrairement à l'approche précédente, l'adaptation dynamique effectue les transformations sur la ressource au cours de son utilisation. Un système d'adaptation automatique intercepte les requêtes des utilisateurs et effectue à la volée des transformations suivant les caractéristiques du contexte d'utilisation actuel. OPERA **[Lemlouma et al., 2001]** et MUSA **[Menkhaus, 2002]** sont des applications où les données sont adaptées dynamiquement aux caractéristiques du terminal utilisé. Elles présentent une adaptation des pages WEB pour différents terminaux en agissant sur leur composition logique, spatiale, temporelle et hypertextuelle. Avec une adaptation dynamique, nous gagnons un temps de développement important puisque la ressource est transformée automatiquement à la demande de l'utilisateur.

Cependant, des problèmes d'adaptation peuvent surgir dans un nouveau contexte d'utilisation inconnu, en fournissant un résultat d'adaptation qui n'est peut-être pas adéquat à ce contexte. En outre, le temps d'attente nécessaire à l'adaptation de la ressource peut gêner l'utilisateur. En effet, l'adaptation dynamique augmente automatiquement la latence de renvoi de l'information demandée par l'utilisateur.

3. Adaptation centralisée

L'adaptation peut concerner une ressource locale à une machine ou distribuée sur plusieurs machines (typiquement un protocole). Dans le cas où la ressource est distribuée, le processus d'adaptation peut être centralisé ou réparti sur les différents pairs où les différentes parties de la ressource ont été stockées.

4. Adaptation distribuée

L'utilisation de cette solution est limitée à des cas particuliers concernant les protocoles de communication. **[Bridges, 2001]** soulève les problèmes de synchronisation de l'adaptation distribuée et propose un protocole dédié pour gérer cette synchronisation.

5. Adaptation Comportementale

L'adaptation peut être qualifiée de comportementale (algorithmique) lorsqu'elle modifie le comportement de la ressource sans affecter sa structure interne. Ceci facilite l'implantation et l'exploitation de l'adaptation. En effet, puisque la structure est la même, la nouvelle ressource reste directement exploitable comme la ressource d'origine. L'implantation est aussi facilitée car cette adaptation n'effectue pas de changements profonds qui demanderaient une ré-évaluation des modifications effectuées. Cependant, cette approche comportementale limite le degré d'adaptation de la ressource. Elle est très utilisée pour l'adaptation de la couche transport à la qualité de service comme dans **[Akan et Akyildiz, 2004]**.

6. Adaptation architecturale

L'adaptation est architecturale lorsque la structure interne de la ressource peut être modifiée en fonction des besoins applicatifs de l'utilisateur. Cette solution est intéressante car elle offre un haut degré d'adaptabilité mais elle reste très difficile à implanter. Elle ne reste possible que si nous disposons d'une connaissance suffisante sur l'architecture de l'application et ses différents composants. L'adaptation architecturale reste restreinte aux applications orientées composants.

Synthèse

Malgré un grand intérêt pour l'adaptation dynamique, l'adaptation statique reste aussi un moyen sûr et efficace pour assurer l'exploitation des ressources dans différents contextes d'utilisation. Pour assurer une adaptation efficace et ouverte à plusieurs contextes, il est important, selon Chaari **[Chaari, 2007]**, d'utiliser les deux approches de façon complémentaire.

L’adaptation centralisée est plus facile à implémenter que la solution distribuée et elle donne des performances meilleures au niveau du temps d'accès à la ressource adaptée.

II.5.3. Adaptation dans les systèmes pervasifs

Principalement, il y a trois genres d'adaptation : adaptation de contenu, adaptation de présentation (ou interface) et adaptation du comportement (service) **[Chaari, 2007]**. Dans ce qui suit nous présentons les techniques existantes pour l’adaptation des données et du comportement des applications.

II.5.3.1. Adaptation de contenu multimédia

Depuis l'apparition des systèmes pervasifs, l'adaptation de contenu multimédia a fait l’objet d’importantes recherches. Plusieurs techniques d'adaptation de données ont été proposées. Ces techniques reposent sur la transformation textuelle, le transcodage d’image ou le traitement vidéo et audio.

Une des questions majeures en terme d’architecture pour le développement d’une fonctionnalité d’adaptation de contenu est de savoir où se réalisent la prise de décision et les opérations d’adaptation. Trois approches générales ont été proposées dans la littérature pour la localisation de la fonctionnalité d’adaptation entre la source qui héberge le contenu et la destination qui le demande :

(i) côté serveur (qui héberge la source de données),
(ii) côté client et
(iii) au niveau d’un intermédiaire situé entre la source de données et le client.

1. Adaptation côté serveur

Dans la première approche, l’adaptation est opérée au niveau du serveur qui héberge la ressource. Le serveur se charge de découvrir les capacités du client et la bande passante disponible et choisit la meilleure stratégie de transformation de contenu. Les solutions côté serveur présentent quelques inconvénients. En effet, les transformations effectuées sur le contenu demandé induisent une charge de calcul et une consommation de ressources conséquentes sur le serveur. Cependant, cette approche reste très adaptée aux situations à faible variabilité vu la simplicité de sa réalisation. Mais elle n'est pas fiable pour des cas où les critères d'adaptation varient fréquemment.

2. Adaptation côté client

Dans la seconde approche, la fonctionnalité d'adaptation réside au niveau du destinataire. Les transformations de contenu nécessaires sont effectuées par le terminal du client, prenant en compte ses capacités matérielles qui sont ainsi directement accessibles. La source initialement fournie demeure inchangée quel que soit le contexte de son utilisation. Le client, à la réception du contenu, transforme celui-ci de manière à pouvoir le présenter ou l'exploiter.

L'adaptation côté client requiert des modifications minimales au niveau du récepteur et peut être extrêmement efficace dans la personnalisation du contenu jusqu'à correspondre exactement et efficacement aux capacités du récepteur. Cette approche est efficace lorsque les caractéristiques de transmission sont moins critiques que les limites d'affichage du terminal. Toutefois, cette solution est mal adaptée aux situations où les propriétés de transmission sont tout aussi importantes que celles de l'affichage. La complexité habituellement élevée des mécanismes d'adaptation freine également l'adoption large de cette approche. En effet, le terminal du client est généralement très limité en capacités de calcul et de stockage **[Chaari, 2007]**.

3. Adaptation intermédiaire

La troisième approche tente de constituer un compromis entre les deux solutions précédentes : la fonctionnalité d'adaptation est placée sur un nœud intermédiaire habituellement qualifié de proxy (proxy) ou passerelle (gateway). Le proxy intercepte la réponse du serveur, décide et effectue l'adaptation du contenu en fonction des préférences de l'utilisateur, des capacités des terminaux et de l'état du réseau, puis envoie le contenu adapté au client.

L'adoption d'une solution à base de proxy présente plusieurs avantages. D'abord, l'adaptation intermédiaire déplace la charge de calcul du serveur de contenu vers le proxy. En second lieu, le proxy peut être positionné au point le plus critique du chemin de données.

La solution basée sur un proxy présente aussi des inconvénients. En premier lieu, elle passe difficilement à l'échelle à cause des coûts de calcul considérables induits par les opérations d'adaptation **[Chaari, 2007]**.

En second lieu, le fait de passer par un intermédiaire pour la transmission d'un contenu soulève des problèmes de sécurité. Le tiers manipulant le proxy doit être digne de confiance

auprès du receveur et de la source. En l'absence d'un tiers de confiance, l'approche "proxy" ne peut pas se charger d'adapter du contenu. De plus, le tiers est susceptible de facturer son service et les ressources sollicitées pour la réalisation de l'adaptation. De ce fait, des mécanismes de comptabilité devraient être intégrés à cette solution afin de mesurer le coût global du processus d'adaptation.

En troisième lieu, la liste des mécanismes d'adaptation de contenu n'est pas exhaustive et pourrait s'enrichir dans un futur proche ; l'intégration de ces nouveaux outils risque de ne pas être possible si le proxy n'est pas extensible. Enfin, placer toutes les fonctionnalités d'adaptation au sein du proxy requiert des unités de calcul puissantes et beaucoup de mémoire.

II.5.3.2. Adaptation de services

Le besoin de l'adaptation de services a été longtemps identifié. Plusieurs approches manuelles et automatiques ont été proposées pour accomplir l'adaptation **[Miraoui, 2009]**. Généralement, la tâche d'adaptation de service consiste à déclencher un service ou à changer la forme d'un service si quelques événements se produisent dans l'environnement de l'équipement informatique.

Les systèmes à services adaptables ont généralement trois parties **[Cremene et al., 2004] :**

- partie modifiable : le service adaptable
- partie monitoring : évaluation en continu du service et de son contexte
- partie contrôle : définit les ordres de reconfiguration en fonction d'une logique qui lui est propre.

Les méthodes conventionnelles d'adaptation de services emploient essentiellement les structures conditionnelles. Une condition se compose de plusieurs variables représentant les entrées de système combinées avec les opérateurs logiques. Une fois qu'une condition devient vraie, le système devrait se comporter d'une façon bien définie. En général, toutes les alternatives doivent être préparées avant que le système ne soit opérationnel, ce qui signifie que la tâche principale du développeur est de prévoir toutes les conditions possibles, chose qui n'est pas évidente !

Un autre inconvénient de ces méthodes est leur aspect statique, qui ne prend pas en considération la nature fortement dynamique d'un système pervasif causée par la mobilité des

utilisateurs et des équipements. En effet, plusieurs paramètres changent soudainement lors du fonctionnement (exécution) du système.

II.5.4. Le processus d'adaptation

En général, nous pouvons décomposer le processus d'adaptation quelque-soit son type en deux étapes successives :

– La prise de décision concernant les opérations d'adaptation à effectuer,

– L'application des actions d'adaptation en utilisant un mécanisme d'adaptation fourni par le système.

Les conditions qui engendrent l'exécution de chaque étape nous permettent de distinguer deux natures d'adaptation : l'adaptation de nature réactive et l'adaptation de nature proactive. Si la prise de décision d'adaptation et son exécution, sont conditionnées par la détection d'une situation pertinente, nous disons que c'est une adaptation réactive.

Alors que si la décision d'adaptation est prise au moment de la détection de la situation pertinente alors que son exécution est conditionnée par un autre évènement, nous disons que c'est une adaptation proactive. D'autres types d'adaptations proactives, consistent à prendre la décision d'adaptation avant la détection d'une situation pertinente, simplement en utilisant l'historique des variations du contexte et un mécanisme d'apprentissage. [Behlouli, 2007]

Conclusion

Plusieurs architectures permettant la conception des systèmes sensibles au contexte ont été proposées. Mais, elles sont presque toutes spécifiques à un domaine d'application et demandent des efforts supplémentaires pour l'adaptation. Les travaux sur l'adaptation dans les systèmes sensibles au contexte se concentrent sur la présentation d'un contenu informationnel sur des dispositifs mobiles.

La conception d'une application sensible au contexte nécessite un support pour les spécifications du contexte, pour la découverte des capteurs qui peuvent acquérir du contexte et agir en fonction des informations contextuelles.

Il est important de séparer l'acquisition du contexte de son utilisation, de stocker des historiques de contexte et de l'interpréter pour avoir des informations de plus haut niveau.

La conception

Ce chapitre est consacré à la présentation de notre proposition. Nous commençons par résumer les travaux similaires qui nous ont inspirés. Ensuite, nous présentons notre approche

pour la modélisation du contexte qui inclut la gestion des préférences de l'utilisateur ainsi qu'un processus pour l'adaptation au contexte.

Nous terminons ce chapitre par les types d'adaptations réalisés.

III.1. Paramètres d'adaptation

Les concepteurs des applications pervasives doivent concevoir des applications pour être utilisées sur des terminaux qui ont des caractéristiques spécifiques (performances très réduites, petites mémoire vive et morte, taille de l'écran réduite, …). De plus, ces applications doivent utiliser le contexte pour réaliser des activités et fournir l'information appropriée aux utilisateurs sans l'interaction humaine.

Donc, dans les systèmes d'information pervasifs, nous devons assurer une adaptation au type du terminal et à l'utilisateur connecté pour garantir une utilisation confortable des applications dans ces nouveaux environnements. Pour réaliser cette adaptation, beaucoup de nouveaux paramètres entrent en jeu :

- paramètres réseau : dans les réseaux sans fil la bande passante est limitée.
- paramètres de l'utilisateur : l'utilisateur est devenu le point central de la conception des systèmes d'information pervasifs. Les concepteurs de ce genre d'application doivent prendre en considération les préférences de l'utilisateur, son emplacement géographique, son profil…
- paramètres du terminal : la diversité des terminaux mobiles influe sur la conception de ces systèmes d'information. Le comportement de ces systèmes doit s'adapter aux capacités matérielles et logicielles de ces appareils.

Tous ces paramètres forment des contextes d'utilisation différents. Dans la plupart des cas, ces paramètres en plus des changements de l'environnement doivent être pris en compte lors de la conception des applications. L'adaptation des applications pervasives comporte :

- l'adaptation de l'information : qui peut porter sur le *contenu* et la *représentation* et prendre en compte les différentes informations du contexte.
- l'adaptation des services : qui consiste à modifier leurs comportements pour qu'ils soient compatibles avec le contexte d'utilisation de l'application.

Dans notre mémoire, nous considérons uniquement l'adaptation de contenu, qui se base sur un ensemble de transformations sur le type, le format et les propriétés des données renvoyées par les services.

Comme il a été présenté dans les chapitres de l'état de l'art, un système pervasif doit surveiller les informations du contexte et selon le changement de ces valeurs, il doit déclencher ou changer la forme du service qu'il offre aux utilisateurs. La première étape dans la conception des applications sensibles au contexte consiste à définir et représenter les informations du contexte. La plupart des travaux de recherche redéfinissent le contexte selon leur champ d'application ou bien considèrent une définition existante qui est la plus adéquate à leurs besoins.

III.2. Travaux similaires

Notre proposition inclut la modélisation du contexte et l'adaptation au contexte ainsi que l'intégration des préférences de l'utilisateur. Pour ce faire, nous nous sommes basés sur des travaux que nous présentons dans ce qui suit:

III.2.1. Définition et modélisation du contexte

Principalement, nous avons deux travaux sur la définition et la modélisation du contexte, à savoir : les travaux de Miraoui **[Miraoui, 2009]** et Chaari **[Chaari, 2007]**.

La définition de Miraoui et Tadj **[Miraoui et Tadj, 2007]**, regroupe les informations de contexte de façon claire. Elle fait une abstraction de la notion du contexte et le voit du point de vue des services offerts par un système pervasif. Les informations de contexte sont :

« Toute information dont le changement de sa valeur déclenche un service ou change la qualité (forme) d'un service ».

Cette définition orientée service du contexte limite l'ensemble des informations contextuelles à celles qui sont nécessaires pour réaliser l'adaptation de services. Cependant, cette définition est utilisée spécialement pour la modélisation du contexte pour des équipements informatiques et non pas pour la conception des applications. Elle ne prend donc pas en compte les autres informations qui peuvent caractériser le contexte, mais ne jouent pas un rôle crucial pour l'adaptation des services.

De plus, Miraoui **[Miraoui, 2009]** propose un modèle de contexte à base d'ontologie, illustré par la figure 1, fondé sur l'idée générale suivante :

« Dans un système pervasif, un équipement informatique fournit un ensemble de services à l'utilisateur et aux applications. Chaque service peut être fourni sous plusieurs formes. Il renferme un ensemble de capteurs intégrés de différents types qui lui permettent de recueillir, avec d'autres capteurs du système pervasif, l'ensemble des informations contextuelles. L'ensemble de ces informations contextuelles déclenchera les services ou changera leurs formes si un sous ensemble de ces informations change ses valeurs. »

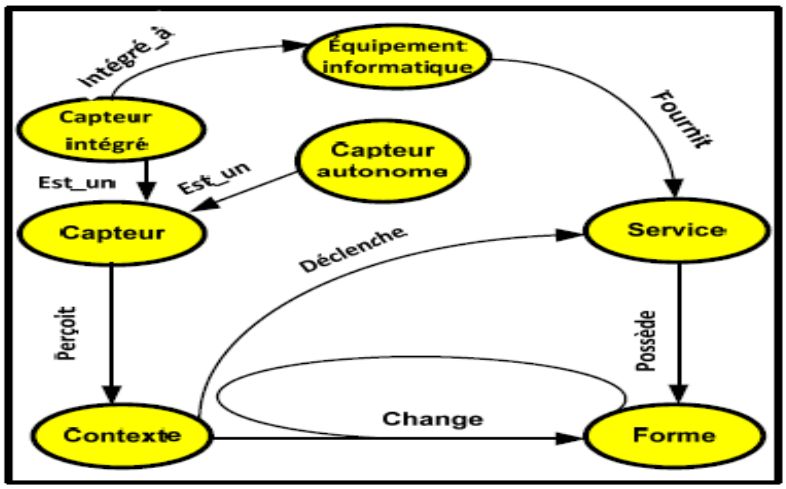


Figure 1. Les éléments de l'ontologie de service [Miraoui, 2009].

Une autre définition du contexte est celle de Chaari **[Chaari, 2007]** :

« Le contexte est l'ensemble des paramètres externes à l'application qui peuvent influer sur son comportement en définissant de nouvelles vues sur ses données et ses fonctionnalités. Ces paramètres peuvent être dynamiques et peuvent donc changer durant l'utilisation de l'application ».

Cette définition du contexte concerne les applications client/serveur. Elle se base sur une étude élaborée sur les caractéristiques des paramètres du contexte afin de faciliter la tâche de leur identification lorsque nous considérons une application donnée.

La figure 2 présente la structure générale du modèle du contexte de Chaari. Il associe un profil « contextProfile » qui décrit les cinq facettes du contexte à chaque session utilisateur *(terminal, communication, utilisateur, localisation, environnement).* Chaque facette peut contenir des paramètres du contexte comme elle peut contenir des catégories pour classer les

paramètres. Chaque catégorie peut contenir d'autres catégories. Nous retenons dans notre proposition la notion de catégorie et de paramètre.

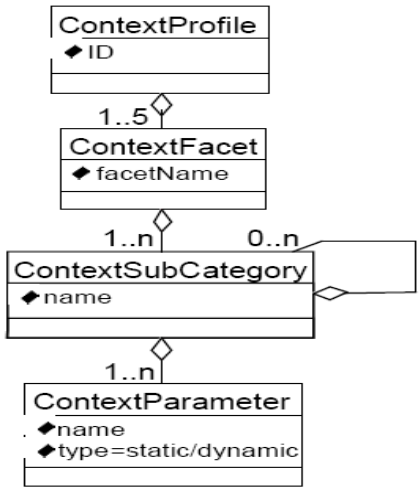


Figure 2. Le modèle général du contexte de Chaari [Chaari, 2007].

III.2.2. Adaptation au contexte

Suite à sa définition du contexte dans un système pervasif, Miraoui **[Miraoui, 2009]** propose la réalisation de l'adaptation des services avec deux opérations (Figure 3) :

- déclenchement automatique d'un service selon le contexte ou
- changement de la qualité d'un service fournis selon le contexte (le service sera fourni sous une autre forme) à cause du changement de la valeur d'un ensemble d'informations.

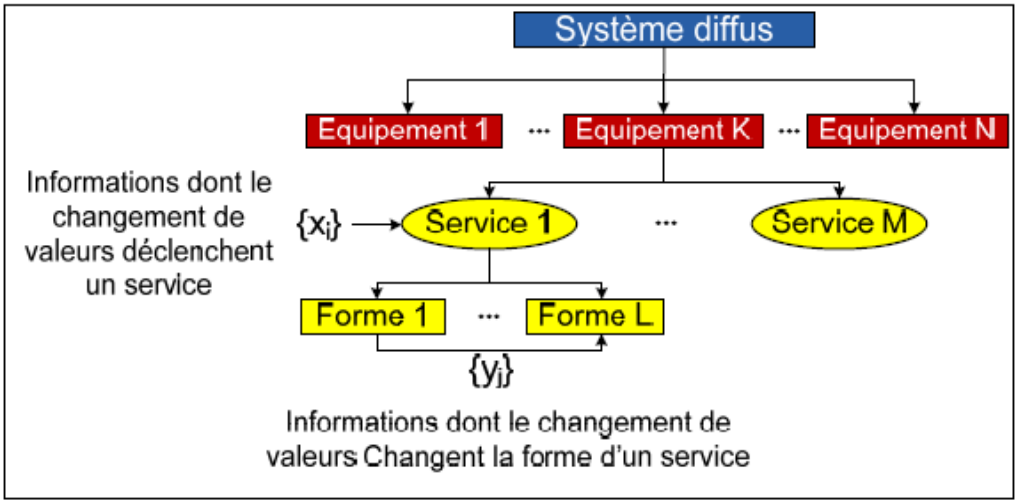


Figure 3. Les composants d'un système pervasif [Miraoui, 2009].

Il présente deux approches d'adaptation dynamique des services pour les systèmes informatiques pervasifs. La première est fondée sur un apprentissage automatique tandis que la deuxième consiste à l'adaptation sensible aux ressources limitées.

Chaari **[Chaari, 2007]** propose une stratégie d'adaptation d'applications au contexte d'utilisation sur trois volets: (i) les services offerts à l'utilisateur, (ii) les données renvoyées par ces services et (iii) leur présentation à l'utilisateur. Il a mis en œuvre cette stratégie en développant une plateforme d'adaptation d'applications au contexte d'utilisation dite SECAS (Simple Environment for Context-Aware Systems).

L'adaptation des services consiste à intercepter les appels vers les services originaux de l'application et à modifier leur comportement à l'aide d'un ensemble d'opérateurs d'adaptation fonctionnelle. L'adaptation des données consiste à transformer ou à remplacer chaque objet multimédia renvoyé par les services de l'application qui n'est pas utilisable dans la situation contextuelle en question. L'adaptation de la présentation se base sur un processus de génération automatique du code de l'interface utilisateur qui garantit l'interaction avec les données et les services adaptés.

III.2.3. Gestion des préférences de l'utilisateur

Dans son travail de recherche, Carrilo et ses co-auteurs **[Carrillo et al. 2006]** proposent *PUMAS* (**P**eer **U**biquitous **M**ulti-**A**gent **S**ystem), un *framework* qui récupère l'information distribuée entre plusieurs systèmes d'information web et/ou accédée à travers différents types de *Dispositifs Mobiles*. Ce qui nous a attiré dans ce travail est le concept de « préférences utilisateur » : un système de gestion de préférences consiste en la capture du contexte d'utilisation de la session et en la sélection des préférences (d'Activité, de Résultat et d'Affichage) qui peuvent être appliquées dans les circonstances de la session en cours.

Nous avons adopté la proposition de Carrillo **[Carrillo et al. 2006]** qui utilise les préférences utilisateur en vue de l'adaptation. Pour satisfaire amplement la demande d'un utilisateur, les systèmes d'information doivent fournir une information juste et pertinente. La pertinence des réponses fournies est une fonction restreinte des souhaits et des désirs de l'utilisateur, c'est-à-dire que cet utilisateur préfère et privilégie une ou plusieurs réponses parmi celles proposées (même si celles-ci sont toutes justes). Chaque utilisateur possède des contraintes spécifiques suivant lesquelles les résultats doivent être adaptés et personnalisés, nous disons que cet utilisateur possède des préférences.

III.3. Présentation générale du modèle du contexte proposé

Inspirée par ces définitions et ces travaux, nous proposons un modèle de contexte, où nous regroupons les informations contextuelles dans quatre éléments :

- Caractéristiques de l'utilisateur
- Caractéristiques du dispositif utilisé
- La communication et
- L'environnement.

Dans notre modèle du contexte (figure 4), nous structurons les caractéristiques du dispositif dans des catégories. Chaque catégorie peut contenir d'autres sous catégories et elle est composée des éléments « Paramètre ».

La représentation proposée du contexte d'utilisation permet que sa définition soit dynamique et extensible puisqu'il est composé d'une ou plusieurs caractéristiques. Ces caractéristiques ne sont pas fixées au préalable et sont définies par le concepteur du système selon ses besoins.

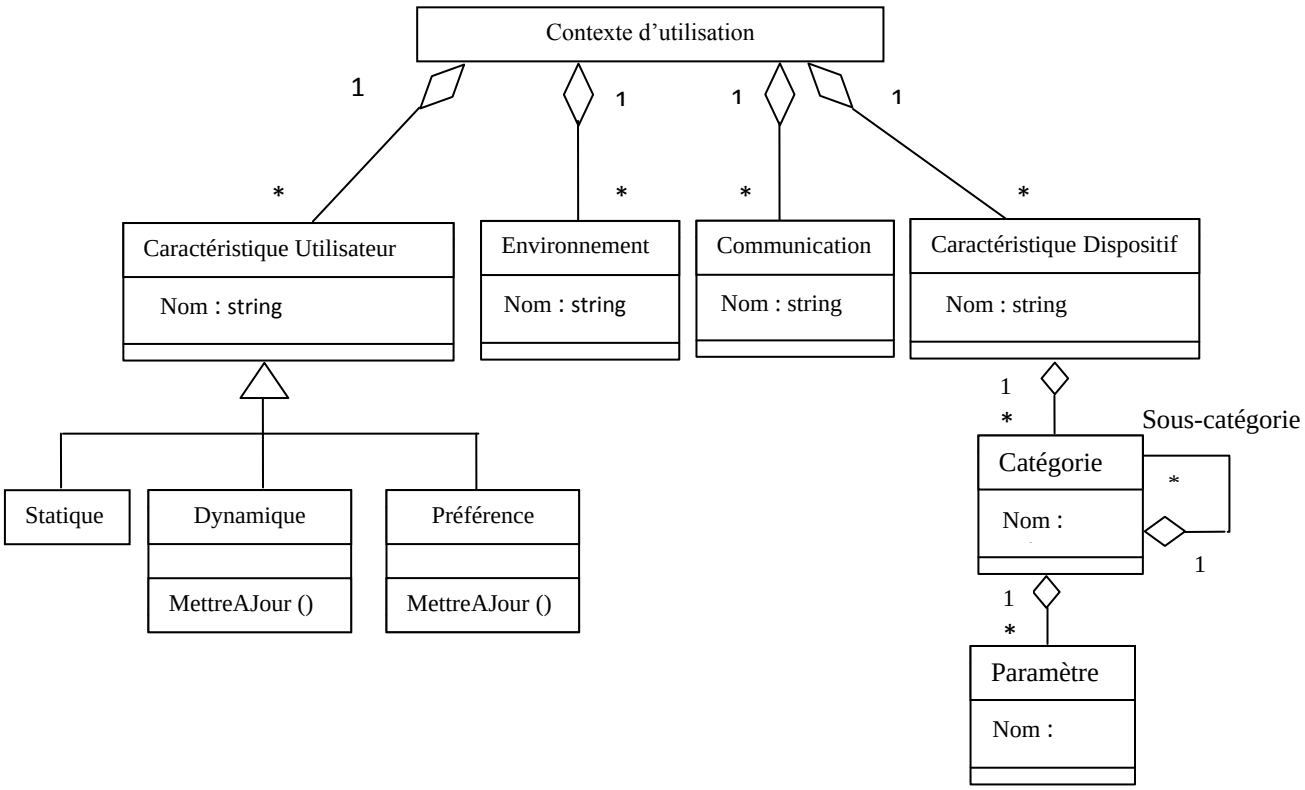


Figure 4. Diagramme de classes de la représentation du contexte.

La figure 4 illustre notre représentation du contexte d'utilisation. Le contexte est décrit par six fichiers XML, chacun pour un élément donné. Nous décrivons les éléments de notre modèle comme suit:

- Les caractéristiques de l'utilisateur : permet de décrire les caractéristiques Statiques de l'utilisateur, les éléments des caractéristiques Dynamiques et des Préférences. En XML, cela est représenté par un élément et par un attribut de cet élément. Par exemple : les caractéristiques statiques peuvent être la langue (élément *langue*) qui a pour valeur 'Français' et le nom de l'utilisateur (élément *nom*) qui a pour valeur 'Mohamed'. Les caractéristiques dynamiques évoluent durant l'utilisation du système, comme par exemple, la localisation de l'utilisateur (élément *localisation*). Les préférences représentent les critères pour lesquels l'utilisateur désire de l'adaptation. Par exemple : l'utilisateur désire que le système prenne en considération ses remarques notées sur son agenda (passage à une pharmacie). L'utilisateur peut aussi désirer que le système déclenche une fonction (*Afficher-Planning*) selon l'heure (élément *temps-fonction).*

- Les caractéristiques du dispositif : Nous respectons la hiérarchie décrite dans le diagramme de classe de la représentation du contexte (Figure 4). Donc, il regroupe les caractéristiques par catégories. Par exemple : la taille de l'écran peut être un élément dans ce fichier (élément *taille-écran*) qui a pour valeur '*800x600*' et qui est un paramètre dans la catégorie (élément *Matériel*). Le réseau supporté par ce matériel peut être aussi un élément de la catégorie réseau (élément réseau) avec comme valeur 'Wi-Fi'.

- la communication : cet élément décrit la connectivité disponible pour communiquer avec l'application. Par exemple : bande passante du réseau sans fil.

- L'environnement : regroupe toute information autre que les trois autres éléments. Elle peut inclure la date, l'heure et toutes les informations du contexte spécifiques à un domaine donné.

Les figures suivantes (Figure 5 et figure 6) donnent des exemples des informations contextuelles associées à un cas d'utilisation.

```
<?xml version="1.0" encoding="UTF-8"?>
<Contexte Id_User="123">
        <Caractéristique_Dispositif>
```

Figure 2. Caractéristiques du dispositif.

Figure 5. Caractéristiques du dispositif. Un utilisateur accède à l'application en utilisant un téléphone mobile de type NOKIA 6230 à connectivité limitée.

```
<?xml version="1.0" encoding="UTF-8"?>

<Contexte Id_User="123">

  <Caractéristique_Utilisateur>

    <Caractéristiques_statiques>
```

Figure 6. Caractéristiques de l'utilisateur.

La section suivante détaille les caractéristiques de l'utilisateur.

III.4. Paramètres de l'utilisateur

Les caractéristiques de l'utilisateur peuvent être statiques, dynamiques ou de préférences. Les caractéristiques statiques de l'utilisateur sont enregistrées lors de sa première connexion au système et restent inchangées pour les utilisations suivantes par exemple : son prénom.

Les caractéristiques dynamiques et les préférences sont enregistrées lors de la première utilisation mais peuvent évoluer soit d'une utilisation à une autre, soit au cours de la même utilisation. Exemple : si nous somme dans un système e-business et si un utilisateur a moins de 25 ans ; l'âge est une caractéristique dynamique, le système peut lui proposer des prix préférentiels. Ce même utilisateur se connecte plus tard, il a maintenant plus de 25 ans, le système ne lui propose plus de prix préférentiels.

D'après cet exemple, nous mettons en évidence que le contexte d'utilisation peut évoluer d'une utilisation à une autre. Un autre exemple peut être la *localisation* de l'utilisateur. Cette caractéristique dynamique peut varier durant une même utilisation si l'utilisateur a pour terminal d'accès un dispositif mobile. Le contexte d'utilisation peut évoluer alors au cours d'une même utilisation de l'application.

Le contexte d'utilisation renferme toutes les conditions d'exécution ou d'utilisation d'une application. Les préférences d'un utilisateur représentent une partie de cet un ensemble global. Une préférence peut être un caractère ou un comportement formulé par l'utilisateur ou par les concepteurs pour exprimer les conditions souhaitées des interactions entre l'utilisateur et les systèmes d'information. L'expression d'une préférence peut se faire de façon explicite (par requête directe) ou bien de façon implicite (par déduction).

Selon Carrillo **[Carrillo et al. 2006]**, les préférences sont classées en trois catégories:

- Préférences d'activité : ce sont les tâches et les fonctionnalités (activités) qu'un utilisateur envisage d'accomplir dans le système et la manière de le faire (séquentielle, concurrente, conditionnelle).
- Préférences de Résultat: spécifient le type et l'ordre des résultats des activités
- Préférences d'affichage : désignent le mode de représentation et la manière dont l'utilisateur souhaiterait que l'information soit affichée.

L'information sur les préférences peut être reconnue par un système selon les trois manières suivantes **[Kassab et al. 2005]** :

- Elle peut être fournie par l'utilisateur.
- Les concepteurs des systèmes d'information peuvent l'avoir définie à travers de profils généraux d'utilisateurs.
- Le système peut la déduire de l'historique de l'utilisateur.

Les préférences de l'utilisateur seront considérées comme des éléments d'entrées qui serviront pour la conception des systèmes d'information. Ceci permet d'assurer la prise en compte de ces paramètres et donc de garantir l'adaptation et la personnalisation des résultats en fonction des objectifs de l'utilisateur.

De plus, lors de l'exécution, les préférences de l'utilisateur vont être un paramètre pour l'adaptation des services selon le choix de l'utilisateur. Dans ce qui suit, nous reprenons la

définition de Carrillo et ses co-auteurs **[Carrillo et al. 2006]** des trois types de préférences (d'*Activité*, de *Résultat* et d'*Affichage*) et donnons leur *portée* (*générale* ou *spécifique*)

III.4.1 Préférences d'Activité

Les préférences d'activité décrivent la manière dont l'utilisateur envisage d'accomplir ses activités dans le système.

Nous définissons le type de la préférence: *générale* ou *spécifique*, la liste des *critères* d'adaptation (par exemple, la localisation et le type de dispsitif) et *l'Activité* que l'utilisateur souhaite accomplir dans le système (par exemple, une consultation).

Cette *activité* est exprimée à travers une *chaîne de fonctionnalités* exécutées de manière séquentielle, concurrente ou conditionnelle.

III.4.2 Préférences de Résultat

Les *Préférences de Résultat* définissent les résultats attendus par un utilisateur à l'exécution d'une ou plusieurs fonctionnalités. Un utilisateur peut établir ses *formats préférés* (par exemple, « *texte* », « *vidéo* » ou « *image* ») pour afficher chaque résultat.

Nous spécifions le nom de la *fonctionnalité* et une liste des couples des résultats et des formats souhaités associés. Un utilisateur peut choisir les résultats de la fonctionnalité (parmi les résultats propres de la fonctionnalité), c'est-à-dire, l'utilisateur peut souhaiter obtenir seulement certains résultats.

III.4.3 Préférences d'Affichage

Les ***Préférences d'Affichage*** décrivent la manière dont l'utilisateur souhaite que son *DM* affiche l'information (par exemple, l'utilisateur désire seulement l'information au format image). Nous les classons par rapport aux formats préférés exprimés par l'utilisateur lors des *préférences de Résultats* et à l'ordre d'affichage de l'information.

Notons que les *préférences d'affichage* peuvent être référencées dans les *préférences de résultat*, constituant dans ce dernier cas, des préférences à appliquer à un format, indépendamment d'un contenu particulier (pour privilégier le format texte pour une session, par exemple).

III.5. Adaptation des applications sensibles au contexte dans les systèmes d'information pervasifs

Nous nous intéressons aux applications qui peuvent proposer des choix d'actions appropriées à l'utilisateur en tenant compte de son contexte d'utilisation, et qui déclenchent une commande ou reconfigurent le système à la place de l'utilisateur, selon les changements des valeurs du contexte. Nous proposons dans la suite un processus de conception d'application sensible au contexte, afin d'adapter les services de l'application aux changements du contexte.

Nous rappelons que l'adaptation se fait sur deux volets :

- Contexte d'utilisation : adaptation des services
- Préférences : adaptation de contenu.

Nous définissons un service comme un processus applicatif autonome. Un service fournit un ensemble de données de sorties suivant un ensemble de paramètres d'entrée.

Une application offre un ensemble de services à l'utilisateur. Certains services peuvent être non exploitables lors du changement du contexte. Par exemple, un service qui retourne un gros volume de données n'est plus utilisable sur un terminal de faible mémoire de traitement. Le comportement de ce service doit être adapté aux limitations matérielles du dispositif de l'utilisateur. Dans ce cas, nous décomposons les données renvoyées par le service en plusieurs blocs, chacun étant directement exploitable par le terminal. Nous remplaçons le service original par un autre service qui renvoie les données bloc par bloc.

Nous définissons l'adaptation fonctionnelle d'une application par le changement du comportement de ses services pour qu'ils puissent être utilisés d'une manière plus conviviale et plus correcte dans le contexte en question.

Dans notre proposition, nous adoptons la définition orientée service de Miraoui pour regrouper les informations de contexte. L'adaptation fonctionnelle consiste à exécuter automatiquement un service (déclenchement) ou à changer la forme du service c.à.d. choisir une autre forme adéquate du service demandé par l'utilisateur (instance). Cette adaptation est réalisée par un module de gestion d'adaptation de services.

Lors de l'exécution d'un service, si les conditions sur les valeurs des informations contextuelles sont valides, le module de gestion du contexte choisit la forme adéquate du service.

Dans ce cas, c'est le concepteur de l'application qui définit les conditions et les actions d'adaptation. Et c'est le module de gestion du contexte qui se charge du déclenchement d'un autre service ou bien du changement de la forme d'un service. L'adaptation peut porter sur les résultats du service ou bien sur l'ordre des fonctionnalités (d'une activité) du service.

Concernant les caractéristiques de l'utilisateur, les paramètres statiques sont consultés au besoin, alors que l'application doit être informée du changement d'un paramètre dynamique.

La figure 7 illustre le mécanisme de gestion du contexte qui va utiliser les informations contextuelles pour adapter le comportement de l'application.

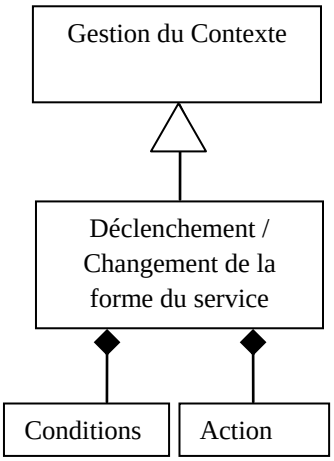


Figure 7. Processus de gestion du contexte

Conclusion

Dans ce chapitre nous avons exposé notre proposition du modèle du contexte en vue de l'adaptation de services.

L'adaptation des applications pervasives comporte *l'adaptation de l'information et l'adaptation des services.* L'adaptation de l'information porte sur le *contenu* et prend en

compte les différentes informations du contexte. Dans notre étude nous avons considéré uniquement l'adaptation de contenu, qui se base sur un ensemble de transformations sur le type, le format et les propriétés des données renvoyées par les services.

Pour l'adaptation de services, elle consiste à modifier leurs comportements pour qu'ils soient compatibles avec le contexte d'utilisation de l'application.

Toute préférence peut ne pas être satisfaite si certaines conditions ne sont pas réunies. Nous devons gérer les conflits entre les préférences utilisateurs et les autres informations du contexte. Le système doit sélectionner seulement les préférences qui peuvent être appliquées selon le *contexte d'utilisation.*

Dans ce mémoire, nous ne gérons pas cet aspect car il ne suffit pas d'un simple filtre au niveau des requêtes tel que c'est le cas dans les systèmes de recherche d'information. Notre proposition concerne les applications qui offrent différents services, il faut prévoir un système plus complexe qui gère ce type de conflit lors de la conception.

Implémentation

Nous décrivons ici le cadre dans lequel notre modèle de contexte a été testé. Nous présentons tout d'abord l'exemple d'application sensible au contexte que nous avons choisi pour la validation, les caractéristiques du dispositif sensé contenir cette application et les outils utilisés pour le développement. Nous commençons par une description de l'application et nous illustrons par un scénario d'utilisation du logiciel.

Lors de la conception du logiciel, nous avons suivi un processus de gestion du contexte visant l'adaptation fonctionnelle de l'application. Cette gestion contient entre autre, la prise en compte des préférences utilisateur. A la fin de ce chapitre, la conclusion permet de recenser les points réalisés dans notre implémentation.

IV.1. Réalisation d'une application sensible au contexte

La plupart des systèmes sensibles au contexte se basent principalement sur la localisation pour définir le contexte d'utilisation. Pourtant, nous avons vu dans l'état de l'art qu'avec peu de moyens supplémentaires, il est possible d'enrichir les applications sensibles au contexte par différentes informations contextuelles. C'est ce que nous prouve le projet Conférence Assistant de Dey, Salber, Abow **[Dey, Salber et al, 1999]**.

Le but de ce projet est de concevoir un organisateur mobile portable pour aider les participants à organiser leur parcours dans les conférences de recherches. Dans notre travail, nous utilisons les spécifications de ce projet pour tester la validité et la généricité de notre modèle du contexte. En suite, nous réalisons la conception et l'implémentation de ce projet selon un processus de gestion d'adaptation de services.

Le dispositif sensé de porter ce projet doit capturer le contexte en mobilité. Un module doit être en fonctionnement quant on le porte, et captent le contexte via des capteurs en permanence pour modifier le comportement de l'assistant dynamiquement. Enfin, cet appareil est toujours allumé pour assurer la continuité de l'évaluation du contexte. On peut utiliser pour cela un assistant personnel dit PDA.

IV.1.1. Les caractéristiques du PDA

Un **PDA** (*Personal Digital Assistant*, littéralement *assistant numérique personnel*, aussi appelé *organiseur*) est un ordinateur de poche composé d'un processeur, de mémoire vive, d'un écran tactile et de fonctionnalités réseau dans un boîtier compact d'extrêmement petite taille.

Utilité du PDA

Le PDA est un ordinateur de poche dont l'usage est prévu originalement dans un but d'organisation. Un assistant personnel fournit donc généralement en standard les applications suivantes :

- Un **agenda**, pour l'organisation de l'emploi du temps, avec des mécanismes de rappel visuels ou auditifs. Les rendez-vous et événements planifiés dans l'agenda peuvent être contextualisés, afin de répondre à des besoins tant professionnels que personnels, grâce à une classification adaptable (*bureau*, *privé*, etc.)
- Un **gestionnaire de tâches** faisant office d'aide-mémoire pour les tâches à effectuer. Une priorité, des dates limites ou des mécanismes de rappel peuvent être affectées à chaque tâche.
- Un **carnet d'adresses** (*gestionnaires de contacts*), permettant d'avoir à tout moment sous la main les coordonnées de ses contacts (numéro de téléphone, adresse postale, adresse de messagerie, etc.).
- Un logiciel de **messagerie**, rendant possible la consultation de ses méls ainsi que la rédaction de nouveaux messages.

Les assistants personnels proposent des outils de bureautique allégés tels qu'un traitement de texte, un tableur, une calculatrice, des visualiseurs pour un grand nombre de formats de fichiers (fichiers PDF, images, etc.). En plus de ces fonctions de base, de plus en plus de PDA proposent des outils multimédias avancés permettant de lire des vidéos (dans les différents formats, y compris le format DivX), des musiques (notamment au format mp3) ou des animations Flash.

Les PDA sont également de plus en plus utilisés pour des usages de géo-localisation, de cartographie et de navigation routière lorsqu'ils sont couplés à un dispositif de géo-localisation (**GPS**, *Global Positionning System*). En effet, pour un faible coût il est possible de

disposer d'un système GPS embarqué très performant permettant une navigation routière à l'aide d'une carte indiquant en permanence sa position, la vitesse et une représentation visuelle de la route (éventuellement en 3D) avec des instructions à l'écran et dictées par une voix de synthèse.

Système d'exploitation

Les PDA possèdent des systèmes d'exploitation dont la définition est adaptée à la résolution d'affichage de l'écran et dont les fonctionnalités correspondent aux caractéristiques de ce type d'appareil. Il existe plusieurs systèmes d'exploitation pour PDA, correspondant la plupart du temps à des types de PDA différents et portés par des constructeurs différents, au même titre qu'il existe des ordinateurs Mac et PC. Les deux principaux systèmes sont :

- **PalmOS**, promu par la société *Palm*.
- **Windows Mobile** ou **Pocket PC** (anciennement *Windows CE*), promu par la société *Microsoft*.

Ces deux systèmes possèdent à peu près les mêmes caractéristiques et les mêmes fonctionnalités avec une prise en main différentes mais surtout des applications incompatibles entre les deux systèmes. Il est à noter qu'il existe des systèmes d'exploitation Linux développés spécifiquement pour les deux types de machines.

Caractéristiques techniques

Lors de l'achat d'un PDA il est notamment important de veiller aux caractéristiques suivantes :

- **poids et dimensions** : Le PDA est destiné à être emporté partout et doit donc tenir dans la main ou la poche. Ses dimensions et son poids doivent donc être choisis les plus petits possible, en gardant à l'esprit le besoin d'ergonomie et de surface d'affichage.
- **autonomie** : L'autonomie du PDA est fonction des caractéristiques de la batterie.
  - **Ni-Cad** (*Nickel / Cadmium*) : type de batterie rechargeable devenu obsolète car souffrant de l'*effet mémoire*, c'est-à-dire une baisse progressive de la charge maximale lorsque celle-ci est rechargée alors qu'elle n'est pas complètement "à plat".
  - **Ni-Mh** (*Nickel / Métal Hybride*): type de batterie rechargeable plus performant que les batteries Nickel-Cadmium.
  - **Li-Ion** (*Lithium / Ion*): type de batterie rechargeable équipant la majorité des ordinateurs portables. Les batteries Li-Ion offrent d'excellentes performances pour un coût modeste. D'autre part les batteries Li-Ion ne souffrent pas de l'effet mémoire, ce qui

signifie qu'il n'est pas nécessaire de vider complètement la batterie avant de recharger l'appareil.

- **Li-Polymer** (*Lithium / Polymère*) : type de batterie rechargeable ayant des performances équivalentes aux batteries Li-Ion mais beaucoup plus légères dans la mesure où le liquide électrolytique et le séparateur microporeux des batteries Li-Ion sont remplacés par un polymère solide, beaucoup plus léger. En contrepartie le temps de charge est plus important et leur durée de vie est plus faible.

Il faut faire attention aux pertes de données lorsque la batterie est totalement vidée.

IV.2. Description de l'application

Les grandes conférences de recherches sont très dynamiques, il y a de nombreuses sessions de conférences en parallèle, sur des sujets très variés impliquant forcément des contextes changeants. L'assistant de conférence est un programme chargé dans un ordinateur portable ou un assistant personnel du participant et qui va lui permettre de décider quelles sessions sont intéressantes pour lui, d'être au courant des activités de ses collègues, d'effectuer des interactions avec son environnement lors du déroulement d'une session. Les éléments du contexte utilisés sont très nombreux : temps, identité, localisation et activité.

Lors de son arrivée à la conférence, le participant reçoit l'assistant sous forme d'un logiciel qu'il va charger dans son dispositif mobile. Les informations qu'il a données lors de son inscription (centre d'intérêt, collègues présents, coordonnées) vont être intégrées dans le logiciel assistant et serviront de contexte de base.

Lorsqu'il n'assiste pas à une session, l'application se présente à l'utilisateur sous forme d'un planning de la conférence avec une présélection grisée des sessions qui peuvent l'intéresser **(**figure 5**)**.

L'utilisateur peut aussi savoir à quelles sessions assistent ses collègues et les notes d'intérêts qu'ils ont accordées aux présentations. Si le participant rentre dans une salle de présentation, l'assistant détecte un changement de contexte et réagit en affichant le nom du conférencier, le titre, le thème et d'autres informations sur cette session. Lors du déroulement de la présentation, l'utilisateur reçoit via l'assistant les diapositives qui sont en train d'être exposées et peut prendre des notes sur celles-ci **(**figure 6**).** Il lui est possible à la fin de la

présentation, lors des questions, de désigner et faire apparaître la diapositive sur laquelle porte sa question sur l'écran du conférencier.

Le participant a la possibilité de noter l'intérêt qu'il accorde à une session de la conférence dans son planning, cette information est répercutée chez ses collègues qui peuvent savoir quelle sont les sessions intéressantes. Il est enfin possible à l'utilisateur, une fois rentré chez lui de retrouver les présentations de la conférence en fonction des sessions auxquelles il a participé, annoté avec ses informations de contexte (Quand est-il arrivé dans la session ? Quelle question a-t-il posée ? Quand est il parti ?) .

IV.3. Conception du logiciel « Assistant de conférence »

L'application « Assistant de conférence » est conçue afin d'explorer le domaine pratique de la sensibilité au contexte. Nous avons utilisé un modèle qui sépare le processus d'acquisition et de représentation du contexte, de l'adaptation.

IV.3.1. Processus de gestion du contexte

Notre modèle fait abstraction de l'étape d'acquisition, et se concentre sur la représentation des informations contextuelles et sur le processus d'adaptation. L'application accède aux informations du contexte via des interfaces graphiques. Ceci permet de cacher la complexité des capteurs physiques.

Donc, la conception consiste à suivre les étapes suivantes :

- Spécification des services de l'application.
- Spécification des formes de chaque service.
- Spécification des informations du contexte : les informations de déclenchement de service et celles qui provoque le changement de la forme d'un service.
- Représentation des informations du contexte.
- Gestion des préférences utilisateurs.

IV.3.2. Spécification des services de l'application

Le tableau ci-dessous énumère les services de l'application, précise les formes de chaque service et les informations du contexte.

Service	Formes	Informations du contexte
Afficher les sessions intéressantes.	- Liste des sessions intéressantes présélectionnées. - Liste avec niveaux d'intérêt des collègues.	- Déclenchement : centres d'intérêts de l'utilisateur, localisation = 'Dehors '. - Changement de formes : localisation = 'salle de présentation', changements d'une valeur des Niveaux d'intérêt.
Afficher les informations sur une présentation.	- Informations sur le conférencier, le thème, le titre et la source de la présentation.	-Déclenchement : localisation = 'salle de présentation' et Heure = 'Heure arrivée'
Afficher les diapos de la présentation en cours.	- Fenêtre séparée. - Fenêtre incluse dans la fenêtre principale du logiciel.	- Déclenchement : localisation = 'salle de présentation' et Heure > 'Heure arrivée'. - Changement de formes : Préférence d'affichage.
Indiquer si un enregistrement Audio/Vidéo est disponible.	- Activer le bouton Audio - Activer le bouton Vidéo	- Déclenchement : localisation = 'salle de présentation' et Heure = 'Heure arrivée'. - Changement de formes : Enregistrement disponible.
Service	**Formes**	**Informations du contexte**
Enregistrer les notes et les attacher aux diapos.		- Déclenchement : localisation = 'salle de présentation' et Changement de la diapo encours.

Envoyer la note d'intérêt de la présentation au serveur.	-Note envoyée et diffusée aux collègues présents.	- Déclenchement : localisation = 'salle de présentation' et Changement de la valeur de la note d'intérêt.
Désigner et apparaître la diapo de la question	-Envoyer le numéro de la diapo au terminal du conférencier. (à travers le serveur)	- Déclenchement : localisation = 'salle de présentation' et Heure = ' Heure des questions'
Enregistrer la présentation avec ses informations de contexte	-Enregistrer les notes sur les questions posées. -Enregistrer l'heure du départ	- Déclenchement : localisation = 'salle de présentation' et Heure = ' Fin de la présentation' -Changements de formes : localisation = 'en dehors de la salle de présentation'

Tableau 1. Les services de l'application.

IV.3.3. Représentation des informations du contexte

Comme outil de modélisation et représentation des informations du contexte, nous avons choisi XML, une solution proposée par le W3C pour le partage d'information (nous avons vu que dans le système pervasif, il y a une communication entre des équipements mobiles et des capteurs de contexte, et avec le serveur d'application).

Dans cette application, beaucoup d'informations contextuelles sont nécessaires pour le bon fonctionnement du système. D'après le tableau des services de l'application, il est clair que les informations de localisation de l'utilisateur et du temps sont très importantes. De plus l'identité, les centres d'intérêt, l'activité de l'utilisateur et ses préférences jouent un rôle dans l'adaptation des services et des résultats de l'application. Le rôle des caractéristiques matérielles du dispositif n'est pas crucial, car comme on a mentionné au début du chapitre, il s'agit d'un PDA dont on connaît approximativement les caractéristiques. Cependant, les

caractéristiques logicielles doivent être connues pour la communication avec les autres équipements mobiles et le serveur.

Notre modèle du contexte nous permet de représenter toutes ces informations comme suit :

- Fichier des caractéristiques statiques de l'utilisateur : le fichier CStatUser.xml décrit les informations sur l'utilisateur, ses centres d'intérêts et ses collègues.
- Le fichier CDynamUser.xml regroupe les éléments dynamiques tels que la localisation de l'utilisateur (élément *localisation*). Les caractéristiques dynamiques évoluent durant l'utilisation du système, ce fichier est très sollicité par l'application.
- Le fichier CPrefUser.xml représente les préférences. En XML, cela est représenté par un élément et par un attribut de cet élément. Les préférences représentent les critères pour lesquels l'utilisateur désire de l'adaptation. Par exemple : l'utilisateur désire que le système déclenche une fonction *(Afficher-Planning*) selon l'heure (élément *temps-fonction)*.
- Le fichier CDispo.xml décrit les caractéristiques du dispositif. Ce fichier respecte la hiérarchie décrite dans le diagramme de classe de la représentation du contexte du chapitre de contributions (Figure III.1). Donc, il regroupe les caractéristiques par catégories. Par exemple : la taille de l'écran peut être un élément dans ce fichier (élément *taille-écran*) qui a pour valeur '*800x600*' et qui est un paramètre dans la catégorie (élément *Matériel*). Le réseau supporté par ce matériel peut être aussi un élément de la catégorie réseau (élément réseau) avec comme valeur 'Wifi'.
- La communication est représentée par le fichier Com.xml.
- Le fichier de l'environnement Env.xml inclut la date, l'heure et toute autre information spécifique du contexte tel que la note d'intérêt.

IV.3.4. Gestion des préférences de l'utilisateur

La gestion des préférences de l'utilisateur consiste à filtrer les résultats des services selon les préférences de l'utilisateur. Les préférences sont représentées comme un élément du contexte. Lors de la conception, le concepteur doit permettre au système d'évaluer ces préférences au regard d'autres informations du contexte courant.

Nous définissons deux types de préférences dans notre application. Il s'agit des préférences générales qui s'appliquent à toute utilisation de l'application c.à.d. les valeurs par défaut de certains paramètres. Le deuxième type regroupe les préférences spécifiques à un utilisateur donné.

Une fonctionnalité est définie par ses paramètres d'entrée et ses paramètres de sortie.

F = NomF (<e_1, e2,…e_n>, <s_1, s2,…, sm>)

1. Préférence d'activité

Dans cette section, nous allons spécifier les tâches et les fonctionnalités que l'utilisateur accomplit dans le système. Il s'agit de la manière de faire (séquentielle, concurrente ou conditionnelle).

La préférence suivante signifie que lorsque l'utilisateur n'est pas dans un lieu de présentation, l'application se présente sous forme du planning des présentations, avec une présélection des sessions intéressantes pour lui.

F_1 = Inscription_Participant(<Nom_User, Liste_Collègues_Prés, Centres_Intérêt>, <espace_réservé>)
F_2 = Consulter_Pres_Interess(<CentresIntérêt, liste_Prés>, <Liste_Prés_Interess, salle, date, heure>)

Préf_Activité (générale, {localisation}, if (localisation<>Lieu_Prés) Then F_2)

L'application permet l'exécution automatique de certains services. Lorsque l'utilisateur rentre dans une salle, un capteur détecte sa localisation, et l'assistant va exécuter les fonctionnalités suivantes selon l'ordre défini par l'utilisateur (ou bien spécifié de manière générale par le concepteur).

F_4 = Afficher_Info_Prés (salle, heure>, <Thème, Titre, Nom_Conférencier, Source_Prés>)
F_5 = Charger_Diapos (<Source_Prés>, <Diapositives>)
F_6 = Prendre_Notes (<Diapos, mémo>, <Fichier_Notes_N°diapo>)
F_7 = Note_Interet (<thème, Nom_User >, <Note>)
F_8 = Diffuser_Note_Interet (<thème, Nom_User, Note, Liste_Collègues_Prés>, <messages_envoyés>)

Préf_Activité (générale, {localisation, heure}, if (localisation=Lieu_Prés) &
(Heure=HPrés) Then F_4 ; F_5 ; F_6 | (F_7 ; F_8))

Dans cet exemple, le concepteur identifie une préférence générale qui précise les tâches qui sont séquentielles et celles qui peuvent être concurrentes.

F_9 = Afficher_Diapo_Question (<N°diapo, Source>, <diapo_affichée>)

F_{10} = Enregistrer_Prés (<Fichier_Notes, video, diapos, questions>, <Prés_Enregistrée, HArrivée, HDépart>)

Préf_Activité (générale, {localisation, heure}, if (localisation=Lieu_Prés) &
(Heure=Hquestion) Then F_9)

Préf_Activité (générale, {localisation, heure}, if (localisation=Lieu_Prés) &
(Heure=FinPrés) Then F_{10})

La dernière préférence d'activité permet aussi une exécution automatique d'un service. C'est le service d'enregistrement de la présentation ainsi que les informations de l'utilisateur. Il s'agit aussi d'une procédure de stockage d'informations contextuelles, auxquelles l'utilisateur peut avoir besoin après la fin de la conférence.

2. Préférence de résultat

Nous spécifions le type et l'ordre des résultats des activités.

Préf_Résultat (générale, F_{10}, < (Présentation, video), (Diapos, Image), (Notes, Texte), (Question, Image)>)

3. Préférence d'affichage

Nous précisons le mode de représentation des données et la manière de leur affichage.

P_1 = Préf_Affichage (générale, Video, {200,300 ; Avi/dat}, P_2)

P_2 = Préf_Affichage (générale, Audio, {mp3, mp4, wmv, rm}, P_3)

P_3 = Préf_Affichage (générale, Image, {600,200 ; jpg, jpeg}, P_4)

P_4 = Préf_Affichage (générale, Texte, {Times, 10, noire, doc, txt}, nil)

P_5 = Préf_Affichage (générale, Texte, {Arial, 11, bleu, doc, txt}, nil)

Nous pouvons aussi lier les préférences de résultat directement aux préférences d'affichage. Par exemple :

Préf_Résultat (générale, F_{10}, < (Présentation, P_1), (Diapos, P_3), (Notes, P_4), (Question, P_3)>)

IV.4. Les outils du développement

1. XML

Beaucoup de développements tournent aujourd'hui autour du web, des assistants de poche, des WAP et autres, il devenait nécessaire de définir une norme universelle de description des données, quelque soit la plate-forme cible. XML répond à cette exigence.

XML est un langage qui permet de structurer des données à l'aide d'une norme de balisage, à l'instar du HTML. Cela permet aux applicatifs d'échanger des données, "formatées". L'intérêt d'XML réside plutôt dans l'existence de bibliothèques qui implémentent les API standards d'XML (DOM, SAX etc.) et ce sur de nombreuses plateformes. Il devient réellement intéressant de concevoir ses paquets de données au format XML, lorsqu'elles doivent être communiquées à un autre service ou système.

XML comporte aussi des outils qui offrent la possibilité de lire et naviguer dans un document XML, comme les parseurs DOM et SAX. Ces parseurs sont des outils implémentant des API standardisées (et/ou en cours de ...) qui offrent des interfaces permettant de parcourir, de rechercher et d'éditer facilement une information au sein d'un document XML.

2. Delphi

Delphi 7 permet de créer, lire et parcourir un fichier XML grâce à ses composants et experts. L'objectif est de créer un paquet de données organisées dans un format souple et reconnu par un maximum de plateformes.

Nous nous appuieront sur l'API standard DOM (Document Object Model) via l'implémentation de Microsoft dans msxml.dll (cette lib doit être enregistrée comme serveur COM au moyen de Regsvr32.exe).

Nous avons besoin dans notre application de lire un fichier XML existant (qui décrit les caractéristiques dynamiques de l'utilisateur, car le contenu de ce fichier peut évoluer au besoin). Delphi 7 propose un expert qui analyse la structure d'un document XML et crée des types interface qui "mappent" les éléments et attributs du document. Il suffit ensuite d'inclure la déclaration de ces types dans les unités Delphi.

Delphi 7 propose des objets OLE pour piloter PowerPoint à partir de notre application Delphi. Dans notre projet, Delphi 7 nous a permis d'afficher automatiquement des diapositives, de récupérer le numéro d'une diapo donnée (la désigner et l'afficher automatiquement) sans intervention explicite de l'utilisateur.

IV.5. Implantation du logiciel

Le logiciel est construit selon notre modèle de contexte décrit dans le chapitre des contributions. Vu que cette application a besoin de communiquer avec d'autres sources d'informations, le système nécessite un serveur central à travers lequel, les instances de l'application chargé dans divers dispositifs mobiles peuvent communiquer et partager les informations contextuelles. Les éléments du contexte partagés sont sauvegardés dans un serveur de contexte accédé au plus haut niveau par l'application.

A l'inscription du participant, le serveur de contexte crée un espace pour l'utilisateur, il va contenir toutes les informations contextuelles du participant pendant sa participation à la conférence. C'est-à-dire il assure certaines fonctions dans le système :

- Acquisition des informations de base (d'enregistrement) : il enregistre les coordonnées, les intérêts et les collègues de la personne.
- Enregistrement des notes (Mémo) : système d'acquisition des notes pendant les présentations et attachement de ces notes aux diapositives correspondantes.
- Sélection des sessions intéressantes : détermine quelles sont les sessions intéressantes pour le participant. il les sélectionne dans le planning de la conférence selon l'information des centres d'intérêts.
- Capture de la localisation de l'utilisateur : enregistre les départs et arrivées du participant dans le lieu d'une présentation.

De même, on trouve des espaces de contextes dans le serveur propres à chacune des sessions, on y trouve toutes les informations sur la présentation de cette section (diapo, vidéo, contexte

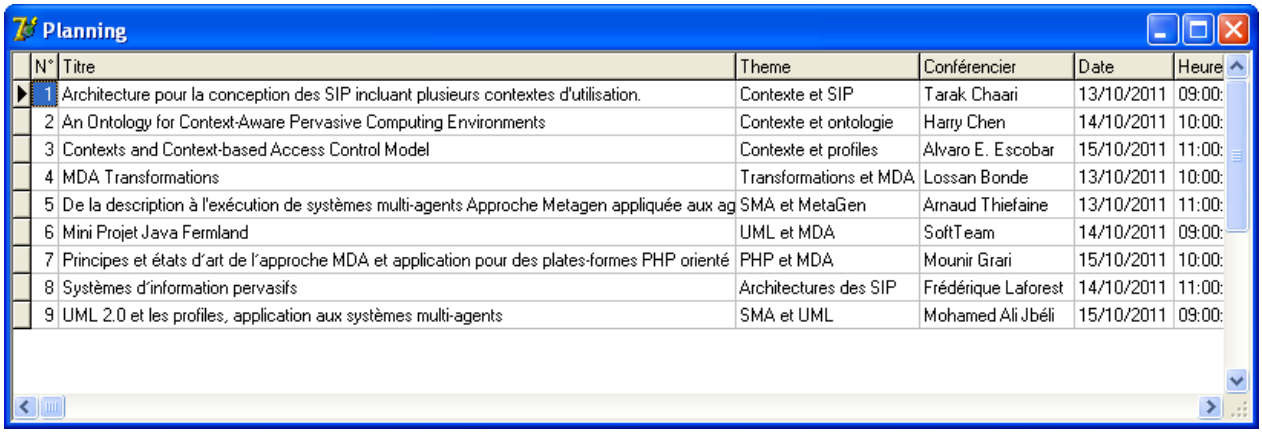
du conférencier) ici aussi le serveur rassemble un certain nombre de services pour les sessions:

- question : acquiert les questions du public.
- contenu : contient les présentations, détecte les changements dans les présentations
- Enregistrement : détecte si oui ou non une vidéo de la conférence est enregistrée

L'application Assistant de conférence ne communique pas avec les capteurs de contexte directement mais passe par le serveur pour récupérer les variables de contextes utilisées pour son exécution.

Dans le menu principal, nous avons commencé par un module de gestion du contexte, il permet la capture du contexte, l'interprétation de ce contexte et son stockage. Nous utilisons les informations captées dans le cas où le changement de leurs valeurs influence sur le comportement de l'application.

Lorsqu'un utilisateur charge le logiciel sur son PDA, il contient déjà la liste des sessions (présentations) de la conférence.



N°	Titre	Theme	Conférencier	Date	Heure
1	Architecture pour la conception des SIP incluant plusieurs contextes d'utilisation.	Contexte et SIP	Tarak Chaari	13/10/2011	09:00:
2	An Ontology for Context-Aware Pervasive Computing Environments	Contexte et ontologie	Harry Chen	14/10/2011	10:00:
3	Contexts and Context-based Access Control Model	Contexte et profiles	Alvaro E. Escobar	15/10/2011	11:00:
4	MDA Transformations	Transformations et MDA	Lossan Bonde	13/10/2011	10:00:
5	De la description à l'exécution de systèmes multi-agents Approche Metagen appliquée aux ag	SMA et MetaGen	Arnaud Thiefaine	13/10/2011	11:00:
6	Mini Projet Java Fermland	UML et MDA	SoftTeam	14/10/2011	09:00:
7	Principes et états d'art de l'approche MDA et application pour des plates-formes PHP orienté	PHP et MDA	Mounir Grari	15/10/2011	10:00:
8	Systèmes d'information pervasifs	Architectures des SIP	Frédérique Laforest	14/10/2011	11:00:
9	UML 2.0 et les profiles, application aux systèmes multi-agents	SMA et UML	Mohamed Ali Jbéli	15/10/2011	09:00:

Figure 1. Planning des présentations de la conférence.

Le participant peut consulter les informations sur les thèmes existants, les conférenciers, ainsi que toutes les informations sur les présentations.

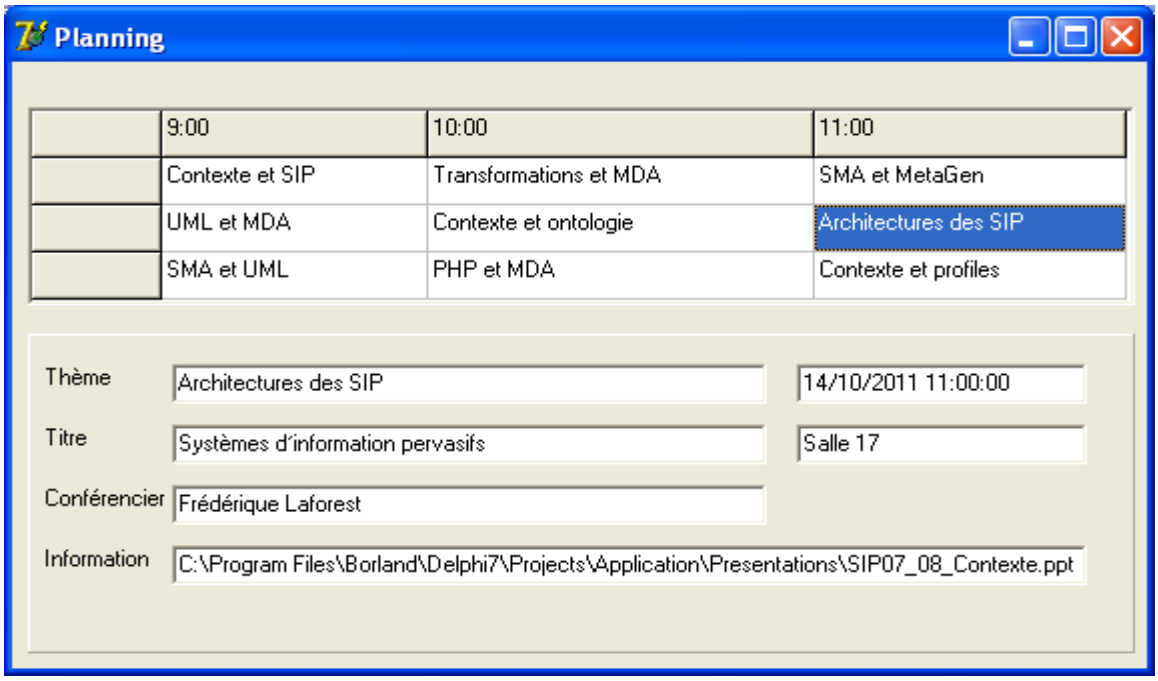


Figure 2. Planning de la conférence avec les informations sur les présentations.

Les informations de base du contexte sont introduites par l'utilisateur lors de son inscription. Il enregistre ses informations personnelles (Nom, prénom, adresse, numéro du téléphone et son adresse E-mail), une liste des centres d'intérêt et la liste de ses collègues présents.

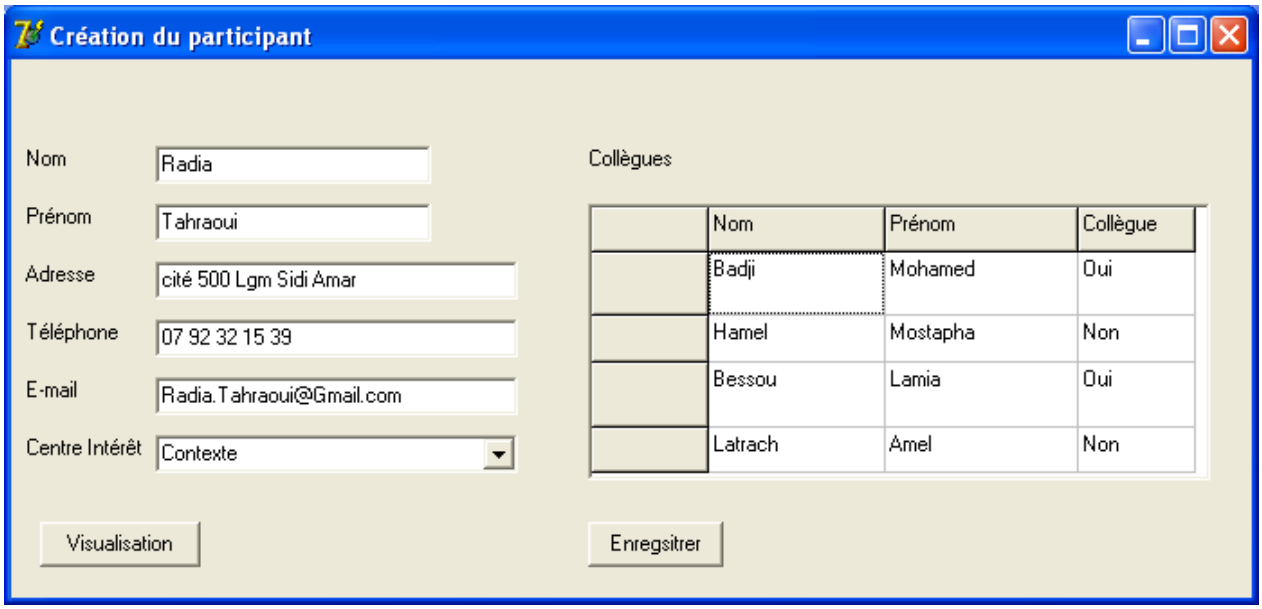


Figure 3. Création du participant.

L'utilisateur peut consulter la liste des personnes inscrites à la conférence et préciser parmi eux ses collègues afin d'être au courant de leurs activité durant la conférence (A quelle session ils assistent et les notes d'intérêt attribuées aux présentations).

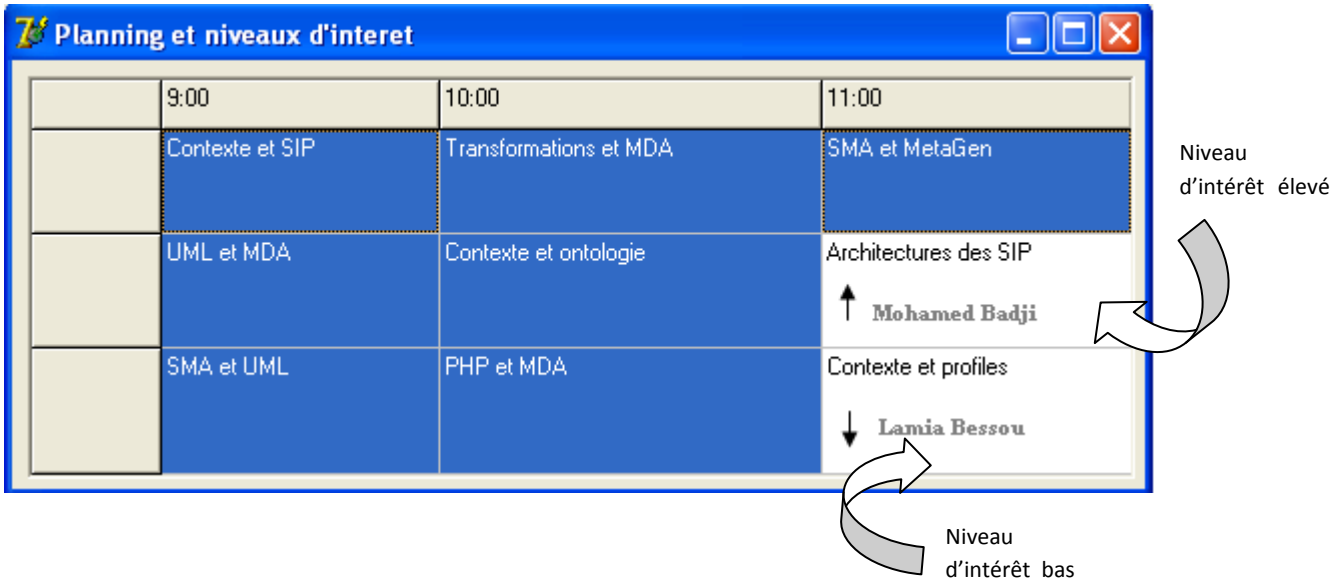


Figure 4. Planning avec les niveaux d'intérêts.

L'information sur les centres d'intérêt servira à sélectionner les sessions intéressantes pour l'utilisateur.

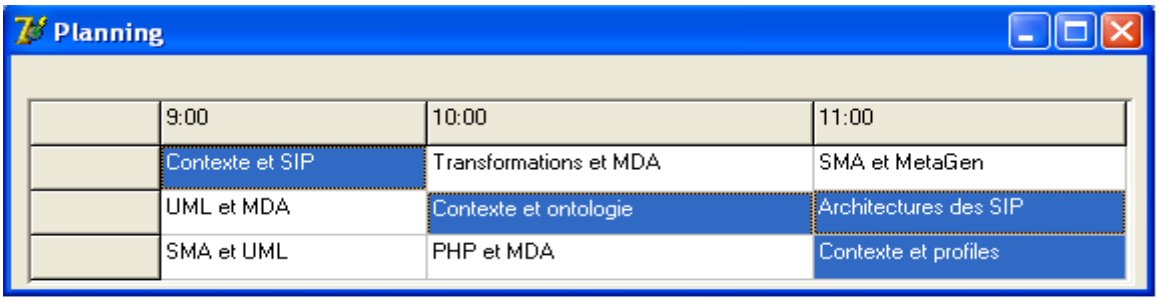


Figure 5. Planning des sessions intéressantes selon les centres d'intérêt de l'utilisateur.

Lorsque le participant prend son dispositif et rentre dans une salle de présentation, l'assistant de conférence détecte le changement de la localisation de l'utilisateur (information du contexte). Le système réagit à cet évènement par le déclenchement d'un service qui va afficher automatiquement les informations sur la session en cours (Nom du conférencier, thème et titre de la présentation et information sur la source, la date et la salle de la présentation).

Le logiciel affiche aussi si un enregistrement Vidéo ou audio de la présentation est en cours. Ceci impacte le comportement du participant, il va prendre des notes brèves ou détaillées selon l'enregistrement disponible.

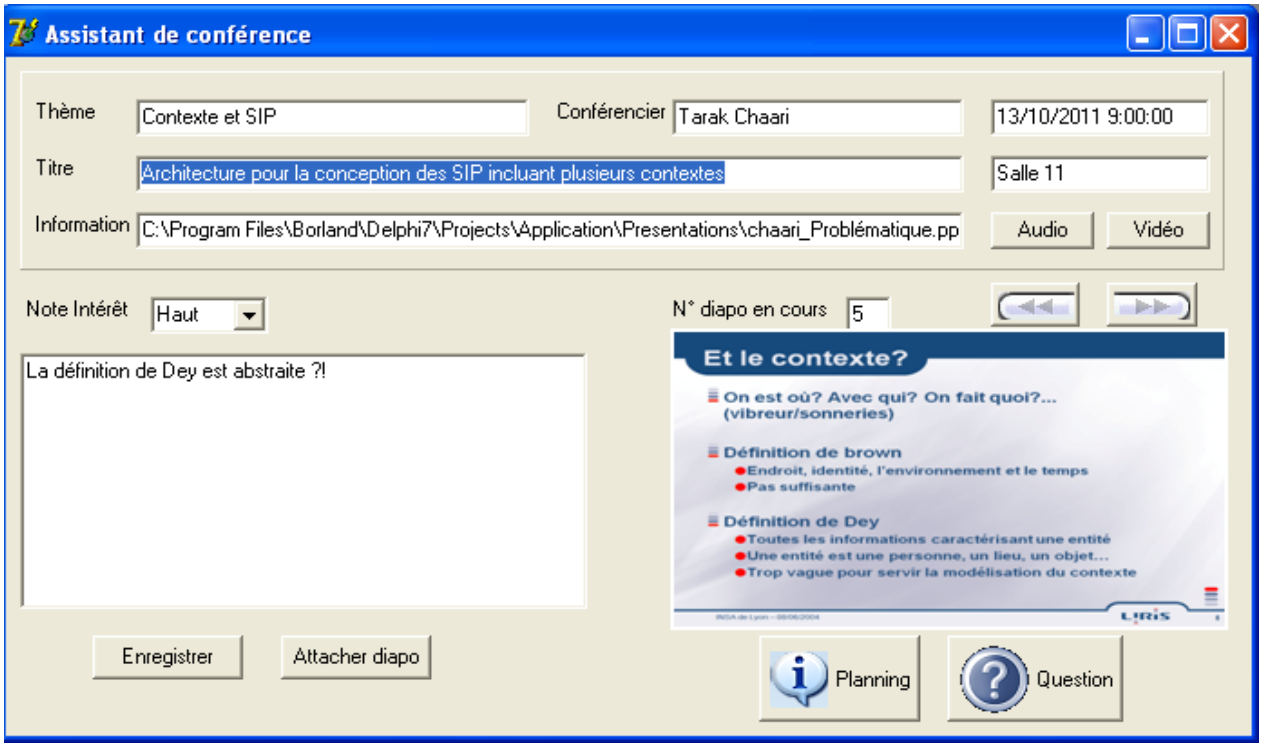


Figure 6. L'interface de prise de notes.

Lors du déroulement de la présentation, l'utilisateur reçoit via l'assistant les diapositives qui sont en train d'être exposées et peut prendre des notes sur celles-ci (figure 6). Les notes vont êtres attachées aux diapos correspondantes.

Lorsque l'utilisateur donne une note d'intérêt à la présentation, cette information est répercutée chez ses collègues (à travers le serveur).

L'assistant de conférence continue à afficher à l'utilisateur les informations du contexte qui changent. L'utilisateur prend ses notes jusqu'à ce que la présentation se termine.

Lors des questions, il lui est possible de désigner la diapositive sur laquelle porte sa question (grâce aux boutons de défilement), et de la faire apparaître sur l'écran du conférencier.

Pour cela, l'assistant de conférence peut communiquer avec l'équipement du conférencier pour contrôler l'affichage de la diapositive en question, ce qui permettra au public présent

dans la salle de la voir. L'utilisateur ajoute les points de la réponse aux notes déjà prises et enregistre toutes ces informations, attachées à la diapo concernée.

A la fin de la présentation, le système enregistre la présentation annotée par les informations du contexte : heure d'arrivée, heure du départ, les questions avec l'heure. Ces informations représentent une base de données, à partir de laquelle, un utilisateur peut consulter tous les mouvements et les activités réalisés au cours de la conférence.

Conclusion

Dans ce chapitre, nous avons utilisé un projet afin de valider notre modèle du contexte et le processus d'adaptation de service.

L'assistant de conférence est une application sensible au contexte qui montre la complexité de la conception de ce genre d'application. Elle utilise différents types d'informations contextuelles aussi bien celles captées et utilisées en temps réel et celles enregistrées pour être utilisées comme historique. Cette application offre trois exemples d'utilisation du contexte : présentation des informations contextuelles à l'utilisateur (le cas des notes d'intérêt attribuées par les collègues), exécution automatique des services (tel que l'affichage des diapos d'une présentation) et l'enregistrement des informations contextuelles pour une utilisation ultérieures.

Lors de la conception du logiciel, nous avons suivi un processus de gestion du contexte. En se basant sur la notion des formes de service et d'informations de déclenchement, il était intéressant de décomposer le projet en services simples.

En suite, nous avons identifié pour chaque service l'ensemble des informations qui influence sur son comportement. La gestion des préférences de l'utilisateur permet à l'application d'être ouverte aux désirs des utilisateurs et autorise l'utilisateur à faire parti du paramétrage des résultats de l'application et donc de participer au processus d'adaptation des services et des données.

Conclusion Générale

Conclusion générale

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

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

Dans ce genre de système, une adaptation de l'application à un ensemble de paramètres (type de terminal, état de connexion, utilisateur connecté…) doit être assurée pour garantir une utilisation confortable. Tous ces paramètres forment un contexte d'utilisation particulier. Dans différents contextes, les utilisateurs accèdent aux mêmes données et aux mêmes services mais reçoivent des réponses qui peuvent être différentes ou présentées différemment ou encore à des niveaux de détails différents.

Le développement d'une application sensible au contexte nécessite de répondre à deux questions principales : (1) Comment concevoir une architecture garantissant l'adaptation au contexte dynamiquement au cours de l'exécution? (2) Comment concevoir l'application elle-même pour qu'elle s'adapte au contexte?

En conclusion, nous pouvons dire qu'un modèle formel et pratique du contexte est encore absent. En plus, dans les applications sensibles au contexte existantes, de gros efforts sont fournis pour définir comment capturer le contexte et le disséminer au système mais il n'y a pas une réponse précise pour savoir comment adapter l'application au contexte. C'est dans cette dernière direction que ce travail s'inscrit.

Dans ce mémoire, nous avons présenté l'importance d'utiliser un modèle adéquat et complet du contexte dans la conception et le développement des applications sensibles au contexte. Un processus d'adaptation des services et des données utilise ce modèle pour assurer une réponse à l'utilisateur selon ses préférences, les caractéristiques de son dispositif et selon les paramètres de son environnement.

Perspectives

- Dans l'implémentation du projet que nous avons choisi, nous faisons abstraction des étapes de capture et de raisonnement sur le contexte. Dans un futur travail, nous allons implémenter ces étapes.
- Toute approche doit être mise en expérience afin de connaître son niveau de complication. Pour cela, nous envisageons de concevoir une application sensible au contexte avec l'approche multi-agent en utilisant notre modèle de contexte et notre processus d'adaptation.
- Notre proposition ne gère pas le conflit entre les préférences de l'utilisateur et les autres informations du contexte d'utilisation. Il faut prévoir un système qui gère ce type de conflit lors de la conception des applications.

Références bibliographiques

[Abowd, 1999] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. 1999. *Towards a Better Understanding of Context and Context-Awareness*. In Proceedings of the 1st international Symposium on Handheld and Ubiquitous Computing, September 27 - 29, 1999 Karlsruhe, Germany. H. Gellersen, Ed. (Lecture Notes In Computer Science, Vol. 1707, pp. 304-307)

[Abowd et al., 1997] Abowd, G., C. Atkeson, J. Hong, S. Long, R.Kooper et M. Pinkerton. 1997. « Cyberguide: A mobile context-aware tour guide ». *Wireless Networks,* vol. 3, no 5, p. 421-433.

[Agoston et al., 2000] Thomas Agoston, Tatsuro Ueda, and Yukari Nisimura, Pervasive computing in a networked world, In Proc. of INET 2000, 18-21 july 2000, Japan.

[Akan et Akyildiz, 2004] Ö B Akan and I F Akyildiz, ATL. *An adaptive transport layer suite for next generation wireless internet.* IEEE Journal on Selected Areas in Communications, June 2004, Vol. 22, N° 5, pp. 802-817

[Albin, 2003] Albin, T.S. 2003. *The art of software architecture design methods and techniques*. Wiley Publishing, Inc.

[Bardram, 2005] Bardram, J. E. 2005. « The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications ». In *the 3rd International Conference on Pervasive Computing (Pervasive 2005)*, sous la dir. de Verlag, Springer. Vol. volume 3468 of Lecture Notes in Computer Science, p. 98-115. Munich, Germany.

[Bauer, 1998] Bauer M., Heiber, T., Kortuem, G. & Segall, Z. *A collaborative wearable system with remote sensing*. Proceedings of the 2nd International Symposium on Wearable Computers (ISWC98), CA : IEEE, 1998, Los Alamitos, pp. 10-17

[Bauer, 2003] Bauer, J. 2003. « Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic ». In (2003). Diplomarbeit.

[Bellucci et al., 2002] Bellucci, S., L. M. Hilty et D. Bütschi. 2002. « Informatique omniprésente: TA-SWISS en analyse les conséquences pour la santé et l'environnement ». In *Société de l'information*. En ligne. <http://www.ta-swiss.ch/www.remain/reports_archive/press_releases/pressemitteilungen2002/PM020924-Pervasive_Computing_f.pdf>.

[Behlouli, 2006] Nabiha Belhanafi Behlouli « Ajout de mécanismes de réactivité au contexte dans les intergiciels pour composants dans le cadre d'utilisateurs nomades » Novembre 2006.

[Biegel et Cahill, 2004] Biegel, G., et V. Cahill. 2004. « A Framework for Developing Mobile, Context-aware Applications ». In *the 2nd IEEE Conference on Pervasive Computing and Communication* (March, 2004). p. 361-365. Orlando, Florida, USA.

[Birnbaum, 1997] Birnbaum, J. 1997. Pervasive information systems. *Commun. ACM* 40, 2 (Feb. 1997), 40-41. DOI= http://doi.acm.org/10.1145/253671.253695 .

[Bouzy et Cazenave, 1997] Bouzy, B., et T. Cazenave. « Using the object oriented paradigm to model context in computer go ». In *Context`97*. Rio, Brazil.

[Bridges, 2001] P G Bridges, W K Chen, M A Hiltunen *Supporting coordinated adaptation in networked systems*. 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), May 2001, Elmau, Germany, pp. 162-168

[Brown, 1996] Brown P.J., «The Stick-e Document: a framework for creating Context-aware applications».Electronic Publishing '96, 1996, p. 259-272.

[Brown et al., 1997] Brown P.J., Bovey J.D., Chen X., «Context-Aware applications: From the Laboratory to the Marketplace», *IEEE Personal Communications*, 4(5), 1997, p. 58-64.

[Carrillo et al. 2006] Angela Carrillo-Ramos, Marlène Villanova-Oliver, Jérôme Gensel et Hervé Martin. « Gestion des préférences utilisateurs pour les Systèmes d'Information ubiquitaires». Actes du 24e congrès INFORSID 2006, Hammamet, 31 mai – 3 juin, 2006, Tunisie.

[Chaari, 2007] Tarak Chaari, « Adaptation d'applications pervasives dans des environnements multi-contextes » Thèse Doctorat en informatique. N° d'ordre 07-ISAL-0058. 2007.

[Chalmers, 2004] Chalmers, M. 2004. « A historical view of context ». *Computer supported cooperative work : CSCW,* vol. 13, p. 223-247.

[Chassot et al., 2006] C. Chassot, K. Drira, K. Guennoun. *Towards autonomous management of QoS through model-driven adaptability in communication-centric systems.* International Transactions on Systems Science and Applications, 2006, Vol. 2, N° 3, pp. 255-264

[Chen, 2003] Chen, H., et al.. *An Ontology for Context-Aware Pervasive Computing Environments*. In Workshop on Ontologies and Distributed Systems (IJCAI 2003), 2003, Mexico.

[Chen, 2004] Chen, H. 2004. « An Intelligent Broker Architecture for Pervasive Context-Aware systems, ». Baltimore County, University of Maryland

[Chen et Kotz, 2000] Chen, G., et D. Kotz. 2000. *A Survey of Context-Aware Mobile Computing Research*. TR2000-381. Dartmouth: Dartmouth College Computer Science

[Chen et al., 2004] Chen, H., T. W. Finin et A. Joshi. 2004. « Semantic web in the context broker architecture ». In *PerCom 2004*, sous la dir. de Society, IEEE Computer. p. 277-286. Orlando, Florida, USA.

[Cheverest et al., 2000] Cheverest, K., N. Davis, K. Mitchel, A. Friday et C. Efstratiou. 2000. « Developing a Context-aware Tourist Guide: Some Issues and Experiences ». In *International conference on Computer Human Interaction (CHI'2000)* (April 2000). p. 17-24 The Hague, Netherlands: ACM Press.

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

[Cremene et al., 2004] M. Cremene et al. « Adaptation dynamique de services », Decor '2004, Grenoble, octobre 2004, pp 53-64

[Dey et al., 1999] K. Dey, Daniel Salber, Masayasu Futakawa and Gregory D.Abowd. *An Architecture To Support Context-Aware Applications*. GVU Technical Report GIT-GVU-99-23. Submitted to the 12th Annual ACM Symposium on User Interface Software and Technology (UIST 99), June 1999.

[Dey, 2001] Dey, A. K. 2001. « Understanding and using context ». *Personal and ubiquitous computing,* vol. 5, p. 4-7.

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

[Dey et al., 2001] Dey, A., G. D. Abowd et D. Salber. 2001. « A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications ». *Human-Computer Interaction,* vol. 16, no 2-4, p. 97–166.

[Dey et al., 1999] Dey A. K., Salber D., Futakawa M., Gregory D., «Abowd An Architecture To Support Context-Aware Applications», GVU Technical Report GIT-GVU-99-23, 1999.

[Dey, Salber et al, 1999] Dey, K. A., Salber, D., Abowd, D. G., Futakawa M. (1999). The Conference Assistant :combining Contextawareness with wearable computing. In Proceedings of the 3rd International symposium on wearable computing. San Fransisco, EtatsUnis, Octobre 1999.

[Fahy et Clarke, 2004] Fahy, P., et S. Clarke. 2004. « CASS - a middleware for mobile context-aware applications ». In *MobiSys Workshop on Context Awareness* (June 2004). p. 304-308. Boston, Massachusetts, USA

[Gicquel, 2010] Pierre-Yves GICQUEL, « Mobilité et sensibilité au contexte » ICI59 - Interaction homme machine 24/11/2010.

[Gu et al., 2004] Gu, T., X. H. Wang, H. K. Pung et D. Q. Zhang. 2004a. « A Middlewar for Context-Aware Mobile Services ». In *IEEE Vehicular Technology Conference* (Spring 2004). Milan, Italy.

[Guha, 1992] R. Guha. *Contexts : a formalization and some applications*. PhD thesis, Stanford, CA, USA, 1992.

[Held et al., 2002] Held, A., S. Buchholz et A. Schill. 2002. « Modeling of context information for pervasive computing applications ». In *the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI '02)* (July 2002). Orlando, Fla, USA.

[Henricksen et Indulska, 2004] Henricksen, K., et J. Indulska. 2004. « Modelling and Using Imperfect Context Information ». In *Workshop Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications (PerCom2004)* p. 33-37. Orlando, FL, USA.

[Henricksen et al., 2005] Henricksen, K., J. Indulska et T. McFadden. 2005. « Modelling Context Information with ORM ». In *OTM Workshops 2005* (2005). Vol. LNCS 3762, p. 626-635. Agia Napa, Cyprus: Springer Verlag.

[Henricksen et Indulska, 2006] Henricksen, K., et J. Indulska. 2006. « Developing context-aware pervasive computing applications: Models and approach ». *Journal of Pervasive and Mobile Computing,* vol. 2, no 1, p. 37-64.

[Hofer et al., 2003] Hofer, T., W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann et W. Retschitzegger. 2003. « Context-awareness on mobile devices - the hydrogen approach ». In *the 36thAnnual Hawaii International Conference onSystem Sciences (HICSS'03)* (January 06-09, 2003). p. 292-302. Hawaii, USA.

[Indulska et Sutton, 2003] Indulska, J. and Sutton, P. *Location management in pervasive systems*. In CRPITS 03: Proceedings of the Australasian Information Security Workshop, 2003, Adelaide, Australia, p. 143–151

[Indulska et al. 2003] J. Indulska, Robinson R., Rakotonirainy A., and Henricksen K. «Experiences in Using CC/PP in Context-Aware Systems**»**. *Proc. 4th Intl. Conf. on Mobile Data Management*, Melbourne, Australia, pp. 247-261, January 2003.

[Kassab et al. 2005] Kassab R., Lamirel J.C, Nauer E,. Une nouvelle approche pour la modélisation du profil de l'utilisateur dans les systèmes de filtrage d'information basés sur le contenu: le modèle de filtre détecteur de nouveautés. Actes de CORIA 2005, Grenoble, France, 9-11 Mars 2005, p 185-200.

[Ketfi et al., 2002] Ketfi, A. and Belkhatir, N. and Cunin, P. Y.. *Adaptation Dynamique, concepts et expérimentations*. In Proceedings of the 15th International Conference on Software & Systems Engineering and their Applications ICSSEA02, December 2002, Paris, France, 8 p.

[Kiciman, 2000] Kiciman E. & Fox, *A. Using dynamic mediation to integrate COTS entities in a ubiquitous computing environment*. In Proceedings of the 2nd International Symposium on Handheld and Ubiquitious Computing (HUC2K). Heidelberg, Germany : Springer Verlag, 2000.

[Korpipää et al., 2003] Korpipaa, P., Mantyjarvi, J., Kela, J. et al.. *Managing context information in mobile devices*. IEEE Pervasive Computing, November/December 2004, Vol. 19, N°6, pp. 21-29

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

[Laforest et Mouël, 2005/2006] Laforest, F., et F. Le Mouël. « Systèmes d'information pervasifs ».<liris.cnrs.fr/frederique.laforest/master/CoursSIP07_08.ppt >.

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

[Lemlouma et al., 2001] T. Lemlouma, N. Layaïda. A Framework for Media Resource Manipulation in an Adaptation and Negotiation Architecture. OPERA project. Grenoble. INRIA Rhône-Alpes, august 2001.

[Linnhoff , 2004] Strang and Claudia Linnhoff-Popien: *A context modeling survey*. In UbiComp 1st International Workshop on Advanced Context Modelling, Reasoning and Management, 2004, Nottingham, pp. 34-41

[Lyytinen et Yoo, 2002] Lyytinen K., Yoo Y., «Issues and Challenges in Ubiquitous Computing», *Communications of the ACM*, 45(12), p. 62-65. 2002

[Matthias, 2007] Matthias Baldauf and Schahram Dustdar. *A survey on context-aware systems.* International Journal of Ad Hoc and Ubiquitous Computing, forthcoming, 2007, Vol.2, N°4, pp. 263-277.

[McCarthy, 1994] J. McCarthy and S. Buvac. Formalizing Context (Expanded Notes). Technical report, Stanford, CA, USA, 1994.

[Menkhaus, 2002] G. Menkhaus. Adaptive User Interface Generation in a Mobile Computing Environment. PhD Thesis. Austria : Salzburg University, 2002.

[Miraoui, 2009] Moeiz Miraoui, **«** Architecture logicielle pour l'informatique diffuse : Modélisation du contexte et adaptation dynamique des services ». Thèse Doctorat en Génie, Ph. Montreal, le 14 juillet 2009

[Miraoui et Tadj, 2007] Miraoui, M., et C. Tadj. 2007b. « A service oriented definition of context for pervasive computing ». In *The 16th International Conference on Computing* (November, 2007).

[Pascoe, 1997] Pascoe, J. « The Stick-e Note Architecture: Extending the Interface Beyond the User ». In *International Conference on Intelligent User Interfaces*. p. 261-264.

[Pascoe, 1998] Pascoe J., «Adding Generic Contextual Capabilities to Wearable Computers», *2nd International Symposium on Wearable Computers*, 1998, p. 92-99.

[Rey, 2006] Gaëtan Rey. « Méthode pour la modélisation du contexte d'interaction » RSTI -SI – 11/2006. Adaptation et contexte, pages 141 à 166.

[Riveill, 2002] Riveill, M.. « Ubiquitous computing: les challenges logiciels ». Université de Nice-ESSI.

[Robinson et al., 2007] Robinson, R., K. Henricksen et J. Indulska.. « XCML: A runtime representation for the Context Modelling Language ». In *The 4th International Workshop on Context Modelling and Reasoning (CoMoRea)*. New York, USA: IEEE Computer Society.

[Rodden, 1998] Rodden T., Cheverst, K., Davies, K. & Dix, A. *«* Exploiting context in HCI design for mobile systems. » In *Proceedings of the Workshop on Human Computer Interaction with Mobile Devices*, 1998, Glasgow, Scotland

[Ryan et Pascoe, 1997] Ryan N., Pascoe J., Morse D., «Enhanced Reality Fieldwork: the Context-Aware Archeological Assistant», V. Gaffney, M. van Leusen, S. Exxon, Computer Applications in Archeology, 1997.

[Ryan, 2007] Ryan, N. 2007. « Contextml: Exchanging contextual information between a mobile client and the field note server ». En ligne. <http://ww.cs.kent.ac.uk/projects/mbicomp/fndConteXtML.html>.

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

[Saha et Mukherjee, 2003] Saha, D., et A. Mukherjee. « Pervasive computing : a paradigm for the 21st century ». *IEEE Computer journal*, (Mars 2003), p. 25-31.

[Satyanarayanan, 2001] Satyanarayanan, M. « Ubiquitous Computing : les challenges logiciels ». *IEEE Personnal communication*, (August 2001), p. 10-17.

[Schilit, 1994] Schilit B.N., Theimer M. «Disseminating Active Map Information to Mobile Hosts», *IEEE Network*, 1994 p. 22-32.

[Schilit et al. 1994] Schilit B.N., Adams N.I., Want R., «Context-Aware Computing Applications», *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA'94),IEEE Press, p 85-90.

[Shanon, 1990] Shanon, B. « What is Context? », *Journal for the Theory of Social Behavior* 1990, Vol.20, pp. 157–166

[Sheng et al. 2005] Sheng Q. Z., and Benatallah B., ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services, the 4th International Conference on Mobile Business (ICMB'05), IEEE Computer Society. Sydney, Australia (2005).

[Strang et al. 2003] Strang T. and Linnhoff-Popien C. «Service Interoperabil-ity on Context Level in Ubiquitous Computing Environ-ments». Intl. Conf. on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (SSGRR2003w), L'Aquila, Italy, January 2003.

[Thevenin *et al.*, 1999] Thevenin D., Coutaz J., «Plasticity of User Interfaces: Framework and Research Agenda», *Proceedings of INTERACT'99*, 1999, p. 110-117.

[Virgilio et Torlone, 2006] Virgilio, R. De, et R. Torlone. « Modeling heterogeneous context information in adaptive web based applications ». In *The 6th international conference on Web engineering* (2006). p. 56-63. Palo Alto, California, USA

[Wang, 2004] H.Wang, X., T. Gu, D. Q. Zhang et H. K. Pung. 2004. « Ontology Based Context Modeling and Reasoning using OWL ». In *Workshop on Context Modeling and Reasoning, In conjunction with the 2nd IEEE Intl Conf PerCom 2004*. Orlando, Florida, USA.

[Want et al., 1992] Want, R., A. Hopper, V. Falcao et J. Gibbons. 1992a. « The Active Badge Location System ». *ACM Transcation on Information Systems* vol. 10, no 1, p. 42-47.

[Want et al., 1995] Want, R., B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis et M. Weiser. 1995. « An overview of the ParcTab ubiquitous computing experiment ». *IEEE Personal Communications,* vol. 2, no 6, p. 28-43.

[Ward et al., 1997] Ward A. J., Hopper A., «A New Location Technique for the Active Office», *IEEE personnal Communications*, 1997, p. 42-47.

[Weiser, 1991] Weiser M. «The computer for the twenty-first century», *Scientific American,* Sept. 1991, p. 94-104.

[Weiser, 1993] Weiser, M. 1993. « some computer science issues in ubiquitous computing ». In *Communications of the ACM*. Vol. 36, p. 74-84.

[Weiser, 1999] Weiser, M. 1999. Some computer science issues in ubiquitous computing. In *Mobility: Processes, Computers, and Agents.* New York, NY : ACM Press/Addison-Wesley Publishing Co., pp. 420-430

[Winograd, 2001] «Architectures for context », *Human-Computer Interaction*, 2001, p.402-419.