Synthèse des méthodes de programmation en informatique contextuelle

Julien Pauty, Paul Couderc, Michel Banâtre

▶ **To cite this version:**

Julien Pauty, Paul Couderc, Michel Banâtre. Synthèse des méthodes de programmation en informatique contextuelle. [Rapport de recherche] RR-5094, INRIA. 2004. inria-00071489

HAL Id: inria-00071489
https://inria.hal.science/inria-00071489

Submitted on 23 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Synthèse des méthodes de programmation en informatique contextuelle

Julien Pauty , Paul Couderc , Michel Banâtre

N°5094

Janvier 2004

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--5094--FR

Synthèse des méthodes de programmation en informatique contextuelle

Julien Pauty , Paul Couderc , Michel Banâtre*

Thème 1 — Réseaux et systèmes
Projet Aces

Rapport de recherche n°5094 — Janvier 2004 — 37 pages

Résumé : L'informatique mobile est un secteur en pleine expansion qui offre tout un ensemble de nouvelles applications. Parmi celles-ci, nous trouvons les applications contextuelles, qui ont pour caractéristique principale de détecter automatiquement la situation courante de l'utilisateur pour y adapter les services qu'elles délivrent. Ce mode de fonctionnement permet de diminuer les interactions entre l'utilisateur et la machine. Il lui offre également la possibilité de se concentrer d'autant plus sur sa tâche courante : faire ses courses, visiter un musée, participer à une réunion ... Dans ce document, nous présentons les différentes méthodes de programmation d'applications contextuelles, qui sont regroupées en deux grandes approches : l'approche logique et l'approche physique.

Mots-clé : informatique contextuelle, informatique mobile,

(Abstract: pto)

* julien.pauty, paul.couderc, michel.banatre@irisa.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

A survey on programming methods for context-aware computing

Abstract: Mobile computing is an innovative field which introduces new applications. Among them, we find the context-aware applications, whose main feature is to automatically detect the current situation of the user to adapt the delivered services. This operating mode allows to reduce the interactions between the user and the computer. It also enables the user to better concentrate on his main task: shopping, visiting a museum, having a meeting ... In this document, we present the different programming methods for context-aware applications, which are grouped in two main approaches: the logical approach and the physical approach.

Key-words: context-aware computing, mobile computing

1 Introduction

La démocratisation de l'informatique mobile et de son principal vecteur qu'est le téléphone portable offre à tout un chacun la possibilité d'accéder à de nouveaux services tels que la téléphonie mobile ou l'accès à internet. Ces services sont globaux. En effet l'utilisation de réseaux cellulaires couvrant la plus grande part du territoire les rend disponibles pour les utilisateurs, quel que soit leur position et déplacements.

Une nouvelle approche, baptisée informatique contextuelle ou ubiquité numérique, envisage un type de service différent, non plus global mais adapté au contexte de l'utilisateur. Elle a pour objectif de lui délivrer des services implicitement, ou tout du moins avec des interactions minimales entre l'homme et la machine. Ici, les services ne sont plus indépendants de la position de l'utilisateur. Au contraire, elle est utilisée pour lui fournir un service approprié à sa situation courante. Les applications fournissant de tels services sont appelées applications contextuelles. Elles ont pour objectif de délivrer à l'utilisateur un service en fonction de son contexte (situation).

Ce nouvel axe de recherche, initié par Mark Weiser [50], a fait l'objet de plusieurs études. Chen et Kotz [6] classifient et présentent tout un ensemble d'applications contextuelles. De leur coté, Touzet et al [47] présentent les différents modes d'accès à l'information en informatique contextuelle. Finalement, dans [10] l'impact de la mobilité sur le développement d'applications contextuelles est étudié.

Dans ce document nous nous intéressons tout particulièrement au problème de la programmation des applications contextuelles. Plusieurs solutions sont possibles, nous les avons regroupées en deux grandes approches que nous appelons approche logique et approche physique. Chacune d'elles fait l'objet d'une étude détaillée dans la suite de cet article. Nous présentons notamment leur modèle de programmation, que nous illustrons par des applications les exploitant.

La partie suivante introduit l'informatique contextuelle. Elle présente précisément la notion de contexte ainsi que chacune des deux approches. Nous poursuivons dans la troisième partie par le détail des applications reposant sur l'approche logique et, dans la quatrième partie, par celles exploitant l'approche physique. Finalement, la cinquième partie compare les deux approches.

2 Informatique contextuelle

2.1 Objectif

L'objectif d'une application contextuelle est de délivrer un service à l'utilisateur en minimisant les interactions avec la machine. En effet, une application contextuelle est utilisée dans des situations où l'utilisateur a déjà une activité, comme visiter un musée ou conduire une voiture. Ses capacités à interagir avec un ordinateur sont donc réduites. Ainsi, l'application cherche à détecter automatiquement la situation de l'utilisateur (contexte) pour lui fournir un service correspondant. Ce étant fait de manière automatique , l'utilisateur se voit

donc déchargé d'une partie de ses responsabilités au profit de l'application. Des interactions réduites lui permettent de se focaliser d'autant plus sur sa tâche principale qu'il n'a pas à détourner sa concentration pour utiliser un ordinateur.

Dans l'idéal, les services sont délivrés sans que les utilisateurs n'aient conscience d'utiliser un ordinateur. Demers fait notamment un parallèle avec l'utilisation d'un téléphone [14]. Lorsque nous voulons parler à quelqu'un de distant, nous prenons simplement le téléphone et l'appelons. Nous ne réfléchissons pas au mode opératoire : décrocher le combiné, composer le numéro ... Nous l'utilisons inconsciemment.

Nous illustrons l'apport de l'informatique contextuelle sur l'informatique traditionnelle par l'intermédiaire d'une application simple. Un guide virtuel est une application qui diffuse aux visiteurs de musées et de lieux touristiques des informations à propos des œuvres ou monuments qu'ils sont en train de regarder. L'approche classique pour réaliser une telle application dispose à coté de chaque œuvre un code numérique. Au cours de sa visite, le visiteur doit rentrer manuellement les codes dans son guide afin d'obtenir les informations correspondantes. Un guide contextuel, lui, détecte automatiquement l'œuvre regardée d'après la position du visiteur. Ainsi, les informations sont déterminées et diffusées à l'utilisateur sans intervention de sa part. Le guide virtuel prend en charge la détection du tableau.

Les applications contextuelles sont variées. Nous pouvons citer notamment les supermarchés électroniques [44], les guides virtuels [1] ou l'enrichissement de documents [49]. Nous en présentons plusieurs en détail dans la suite de ce rapport. Pour une classification d'applications contextuelles plus détaillée, le lecteur pourra se référer au document de Chen et Kotz [6]. Pascoe présente également une classification des applications basée sur leur mode d'exploitation du contexte [36].

2.2 Contexte

Le contexte est une notion centrale en informatique contextuelle et sa définition est une étape fondamentale dans la conception d'une application. Dans cette partie nous présentons de manière précise la notion de contexte ainsi que quelques techniques permettant de le capturer.

2.2.1 Relativité du contexte

En consultant plusieurs dictionnaires, nous constatons que la notion de contexte est avant tout une notion linguistique ou temporelle. Le *petit Larousse illustré* définit le contexte ainsi :

- ensemble du texte qui précède ou suit une phrase, un groupe de mot, un mot ;
- ensemble de circonstances qui accompagnent un événement : replacer un fait dans un contexte historique.

Ces définitions montrent que le contexte d'un élément (phrase, mot, événement ...) n'est pas uniquement caractérisé par sa position, mais bien par sa position relativement à d'autres éléments de même type. Par exemple, les États Généraux se sont déroulés avant la prise de la Bastille et la déclaration des droits de l'homme a eu lieu après. Le contexte

historique de la prise de la Bastille n'est pas représenté par le 14 Juillet 1789, mais par l'ensemble des faits qui se sont déroulés autour de cette date.

Le mot clé pour définir le contenu du contexte est "entourer". Le contexte représente ce qui entoure : les événements précédant et suivant, les mots entourant un autre mot. De même, lorsque nous donnons notre position à une autre personne, le plus souvent nous lui indiquons près de quels bâtiments nous nous trouvons et rarement le nom de la rue et le numéro ; nous décrivons ce qui nous entoure.

2.2.2 Contexte et informatique contextuelle

Dans le cadre de l'informatique contextuelle, plusieurs définitions ont été proposées. Chen et Kotz [6] définissent le contexte comme étant l'ensemble des propriétés physiques qui soit déterminent le comportement de l'application, soit sont pertinentes pour l'utilisateur et lui sont présentées lorsqu'elles évoluent. La définition de Dey et Abowd [15] est la suivante : le contexte est représenté par toute information qui peut être utilisée pour caractériser la situation d'une entité, une entité étant une personne, un lieu ou un objet, ayant rapport avec les interactions entre l'utilisateur et l'application.

Ces définitions permettent de représenter le contexte par des informations physiques comme la position ou la température extérieure, mais aussi la proximité de personnes ou d'objets. De nombreuses applications utilisent d'ailleurs la position géographique comme seule et unique information pour représenter le contexte de l'utilisateur. Nous obtenons une notion de contexte très large pouvant englober toute information concernant l'environnement ou l'utilisateur.

Les contextes correspondant à ces définitions ne mettent cependant pas forcément en valeur l'aspect relatif. Selon nous, la position de l'utilisateur, ou toute autre information de ce type, ne représente pas le contexte, mais est un moyen pour l'obtenir. Un changement de contexte implique un changement de position mais la réciproque n'est pas vraie. Par exemple, dans le cas d'un guide virtuel, le contexte de l'utilisateur correspond aux œuvres ou monuments à proximité. Ce contexte est bien dépendant de la position de l'utilisateur, cependant, si elle change, le contexte ne changera pas forcément pour autant. Si le visiteur marche le long de l'allée centrale d'une église, sa position change et pourtant l'église reste proche de lui et fait toujours partie de son contexte.

2.2.3 Vers un contexte unifié

Nous proposons maintenant une nouvelle définition à la fois plus proche de celles issues des dictionnaires et plus précise que les précédentes. Elle présente l'avantage de définir clairement la notion de contexte.

Dans la suite de ce document, nous nous référerons à cette nouvelle définition sous le terme de contexte *unifié* et à la définition habituelle sous le terme de contexte *non unifié*. Nous utilisons le terme unifié car cette définition est indépendante de l'espace considéré : espace physique, temps, texte ...

Une approche informelle Considérons un espace[1], au sens mathématique du terme, peuplé d'éléments. La nature des éléments dépend de l'espace. Pour l'espace physique, ce sont des points et pour un texte des phrases ou des mots. Nous cherchons à déterminer le contexte d'une entité, qui correspond le plus souvent à l'utilisateur, évoluant au sein de l'espace considéré.

Si nous nous référons aux premières définitions que nous avons données pour la notion de contexte (cf 2.2.1), le contexte d'une entité est représenté par les éléments qui l'entourent. Il est donc local et dépend de la position de l'entité considérée. Nous pouvons relier la notion "d'entourer" à une notion de proximité. Les éléments qui entourent une entité sont les éléments qui sont proches d'elle. Nous considérons un élément comme proche d'une entité lorsque la distance qui les sépare est inférieure à une certaine limite. Ainsi, le contexte d'une entité est représenté par l'ensemble des éléments situés à une distance inférieure à un maximum donné. Par exemple, si nous considérons l'espace physique peuplé de personnes et fixons la distance maximum à dix mètres, le contexte d'un individu est représenté par l'ensemble des personnes situées à moins de dix mètres de lui.

Définition Considérons un espace A, peuplé d'éléments de même type, dans lequel évolue une entité E. Soit $C(E)$ un sous-ensemble d'éléments de A représentant le contexte de E. Nous appelons élément contextuel, un élément de A susceptible d'appartenir à $C(E)$. Soit A_C l'ensemble des éléments contextuels, nous avons $A_C \subset A$. De plus, $C(E)$ n'étant constitué que d'éléments contextuels, nous avons donc $C(E) \subset A_C$.

Le contexte de E est représenté par l'ensemble des éléments de A_C situés à une distance de E inférieure à un maximum L:

$$C(E) = \{e_1,e_2,\ldots,e_n\}/\forall i,d(E,e_i) \leq L \wedge e_i \in A_C \quad (1)$$

$d(E,e_i)$ étant une fonction donnant la distance entre E et e_i.

Nous définissons maintenant le sous ensemble $S(e)$ de A, e étant un élément ou l'entité :

$$S(e) = \{e_1,\ldots,e_n\}/\forall i,d(e,e_i) \leq L \quad (2)$$

$S(e)$ définit une sphère de rayon L et de centre e. Pour un espace à deux dimensions $S(e)$ définit un cercle et un intervalle centré sur e pour un espace à une dimension. Nous appelons le sous ensemble $S(e)$ zone contextuelle de e. $S(e)$ est définie par rapport à la position de e. Si e est mobile, sa zone contextuelle se déplace donc avec lui.

A partir de la définition (2) nous pouvons réécrire la définition (1) de deux manières différentes :

$$C(E) = \{e_1,e_2,\ldots,e_n\}/\forall i,e_i \in S(E) \wedge e_i \in A_C \quad (3)$$

$$C(E) = \{e_1,e_2,\ldots,e_n\}/\forall i,E \in S(e_i) \wedge e_i \in A_C \quad (4)$$

1. Dans cette partie nous considérerons exclusivement l'espace physique pour illustrer la définition d'un contexte unifié. Nous présenterons ensuite son application à d'autres espaces.

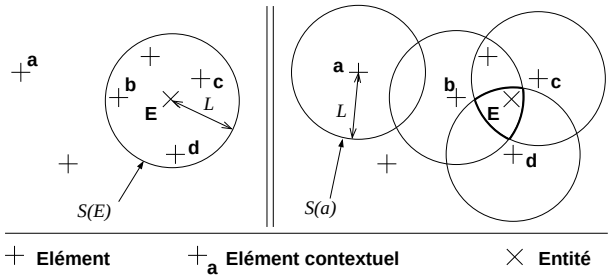

FIG. 1 – *Exemple d'un contexte unifié*

Avec la définition (3) nous définissons une sphère autour de E et $C(E)$ contient l'ensemble des éléments contextuels situés à l'intérieur de cette sphère. Avec la définition (4) nous définissons une sphère autour de chaque élément contextuel. $C(E)$ contient donc l'ensemble des éléments pour lesquels E est comprise dans leur sphère. La figure 1 présente les deux possibilités à l'aide d'un exemple. L'espace physique est peuplé de six éléments, dont quatre sont des éléments contextuels, et d'une entité E. Nous avons $A_C = \{a,b,c,d\}$ et $C(E) = \{b,c,d\}$. La partie gauche illustre la définition (3) et celle à droite, la définition (4).

Avec la définition 2 la limite L est constante quelle que soit la direction considérée. Autrement dit, nous ne pouvons définir que des sphères. Pour palier à cette limitation nous proposons de définir L comme étant une fonction de la position de e_i. Nous avons donc :

$$S(e) = \{e_1, \ldots, e_n\} / \forall i, d(e,e_i) \leq L(e_i) \tag{5}$$

Maintenant, la fonction L renvoie une distance qui est fonction de la direction donnée par $\overrightarrow{ee_i}$. Ceci permet de définir des formes quelconques, telles que des cubes, cônes, cylindres ...

La précédente définition de $S(e)$ associe la même forme à tous les éléments contextuels. Il serait intéressant de pouvoir avoir une forme différente pour chaque élément. Nous redéfinissons $S(e)$ ainsi :

$$S(e) = \{e_1, \ldots, e_n\} / \forall i, d(e,e_i) \leq L_e(e_i) \tag{6}$$

avec L_e représentant la distance limite, associée à e, pour qu'un élément appartienne à $S(e)$. En employant cette nouvelle définition avec la définition (4) nous pouvons donc associer une forme différente à chaque élément. Notons qu'elle ne modifie pas le sens de la définition (3) car celle-ci associait déjà implicitement $L(e_i)$ à E.

Avec ces définitions il est possible de construire des contextes unifiés selon deux points de vue : par rapport à l'entité, avec la définition (3); par rapport aux éléments, avec la définition (4). Dans la suite de ce document, nous faisons implicitement référence à la définition (4) lorsque nous parlons de contexte unifié. Dans le cas contraire, nous précisons explicitement que nous utilisons la définition (3). En ce qui concerne $S(e)$, nous utilisons dans les deux cas la définition (6).

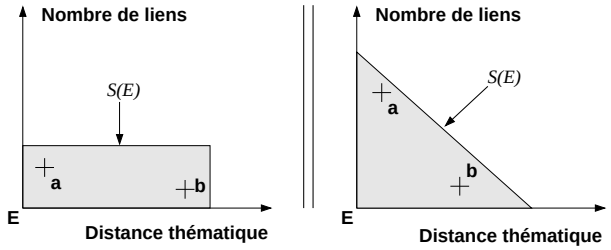

FIG. 2 – *Exemples de zones contextuelles pour le Web*

Mise en œuvre Pour mettre en œuvre un contexte unifié dans une application, celle-ci doit définir l'espace considéré, les éléments et zone contextuels, ainsi que la méthode permettant de déterminer si l'entité (respectivement un élément) est à l'intérieur de la zone contextuelle d'un élément (respectivement de l'entité). Nous illustrons ceci par trois exemples.

Si nous considérons l'espace physique, les éléments qui le peuplent sont des points. Les éléments contextuels sont des données réparties dans l'espace, éventuellement embarquées sur des personnes ou des objets et l'entité est un individu. Les données et l'entité sont bien sûr associées au point de l'espace qui coïncide à leur position. La zone contextuelle d'une donnée est représentée par une portion de l'espace physique. L'appartenance à une zone contextuelle peut être obtenue en utilisant un système de localisation, comme le GPS, en évaluant la distance entre les points associés à la donnée et à l'entité. L'appartenance à une zone contextuelle peut aussi être obtenue par l'intermédiaire d'une interface réseau sans fil (cf 2.2.4).

Nous considérons maintenant le Web comme étant l'espace dans lequel se déplace l'entité. Les éléments contextuels sont des pages Web. La position de l'utilisateur est représentée par la page qu'il consulte. Pour cet exemple, nous munissons le Web de deux dimensions. Selon la première dimension nous mesurons la distance thématique entre deux pages. Cette distance peut être évaluée par des techniques similaires à celles employées par les moteurs de recherche [4]. Selon la seconde dimension nous mesurons le nombre de liens hypertexte à suivre pour atteindre une page depuis une autre. Dans cet espace à deux dimensions, la définition (6) permet de créer des zones contextuelles de forme quelconque. Nous décidons d'utiliser la définition (3) pour construire $C(E)$. La figure 2 présente deux exemples de zones contextuelles. Le rectangle implique une limite constante sur chacune des dimensions. Le triangle impose qu'une page éloignée selon une dimension doit être proche selon l'autre.

Pour l'espace temporel les éléments sont des dates. Les éléments contextuels sont des évènements précis et les zones contextuelles des intervalles de temps s'étendant autour des évènements. L'entité est associée à la date courante et son appartenance à un intervalle est évaluée mathématiquement. Elle peut être un utilisateur ou un évènement dont on veut déterminer le contexte. Avec un tel espace, la définition (6) permet de définir des intervalles

non symétriques. Les deux directions possibles sont le passé et l'avenir. Selon l'importance de l'évènement, la zone contextuelle peut être plus ou moins grande. Le contexte de l'entité est constitué d'un ensemble d'évènements.

2.2.4 Capture du contexte

L'opération consistant à extraire le contexte de l'environnement physique est appelée capture du contexte. Si nous considérons les contextes non unifiés, composés de données physiques telles que la position ou la température, la capture est réalisée directement par des capteurs adaptés. La position étant l'information la plus utilisée, plusieurs solutions ont été développées, la plus connue étant le GPS. Hightower et Borriello les ont recensées et classifiées dans un document de synthèse [23]. Des capteurs dédiés à d'autres informations, telles que l'accélération, la température ou le cap sont également disponibles, leur utilisation dans l'informatique contextuelle étant par ailleurs présentée dans [45]. La reconnaissance d'images est également utilisée pour extraire des propriétés physiques de l'utilisateur, comme par exemple l'orientation de son regard [11].

Détecter le contexte dans le cas d'un contexte unifié revient à détecter à l'intérieur de quelles zones contextuelles l'on se trouve, pour en déduire ensuite les éléments correspondants. Par exemple, considérons l'espace physique, peuplé de personnes disposant chacune d'un terminal sans fil. Nous décidons que la zone de communication du terminal représente la zone contextuelle d'une personne. Si l'utilisateur peut communiquer avec un autre individu alors il se trouve à l'intérieur de sa zone contextuelle. Son contexte unifié est donc représenté par l'ensemble des personnes avec lesquelles son terminal peut communiquer. Capturer son contexte revient donc à détecter les terminaux avec lesquels il peut communiquer.

2.3 Deux approches pour la programmation des applications

Nous introduisons les deux approches via l'application de guide virtuel. Une première possibilité de réalisation, pour une telle application, consiste à équiper le terminal utilisateur d'une interface réseau sans fil et d'un mécanisme de localisation qui permet d'obtenir sa position dans le musée. L'ensemble des informations liées aux œuvres est stocké dans un serveur de données. A partir de la position de l'utilisateur, le serveur est capable de déterminer les données à lui transmettre. Le programme exécuté par le guide est un cycle de capture de la position et d'interrogation du serveur de données.

La seconde approche exploite un contexte unifié. Pour cela, les données sont directement embarquées sur les œuvres auxquelles elles sont associées, via un calculateur sans fil. La portée de communication du calculateur correspond à la zone contextuelle des données qu'il embarque. De ce fait, si un guide peut communiquer avec un calculateur, alors il se trouve à l'intérieur de la zone contextuelle des données correspondantes. Autrement dit, d'après la définition (4), si le guide de l'utilisateur peut communiquer avec un calculateur, alors les données embarquées par ce calculateur seront incluses dans le contexte. Le guide se contente ensuite d'afficher le contenu du contexte. Cette fois ci, l'application devient un cycle de détection des calculateurs et d'affichage du contexte correspondant.

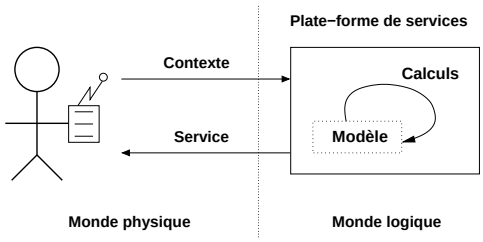

FIG. 3 – *Approche logique*

Avec la seconde méthode nous considérons que les données sont stockées dans l'espace physique, d'où le nom approche physique. Par opposition à cette approche, la première est appelée approche logique. Dans ce cas là, les données sont dissociées de l'espace physique et la position de l'utilisateur est nécessaire pour extraire les informations pertinentes.

2.3.1 Approche logique

Principe En observant la structure d'un guide virtuel nous pouvons remarquer que la délivrance du service se divise en trois étapes : la capture du contexte, l'exploitation du contexte pour déterminer les données à envoyer et l'envoi en lui-même. La phase de calcul située entre la capture et l'envoi caractérise l'approche logique.

D'une manière générale la phase de calcul exploite le contexte pour délivrer le service. Elle peut également faire appel en plus à une modélisation du monde physique, comme par exemple une cartographie des lieux. La machine qui l'exécute est appelée plate-forme de services. Si nous revenons à l'exemple du guide virtuel, le contexte correspond aux coordonnées de l'utilisateur et le modèle est représenté par une base de données qui met en correspondance coordonnées et informations. La plate-forme de services est la machine qui contient la base de données, reçoit les coordonnées et renvoie les données.

Une application contextuelle programmée avec l'approche logique suit donc le modèle présenté sur la figure 3. Elle est constituée d'un ensemble de nœuds mobiles qui envoient leur contexte vers la plate-forme de services. En retour, celle-ci leur délivre un service qui est fonction de ce contexte. La plate-forme de services est constituée d'un programme qui calcule le service et s'appuie éventuellement sur un modèle du monde physique.

Architecture matérielle Une application logique est composée de plusieurs éléments : terminal utilisateur, infrastructure éventuelle pour capturer le contexte, plate-forme de services. La combinaison de ces différents éléments permet d'obtenir différentes configurations pour les applications exploitant l'approche logique. Tout d'abord la plate-forme de services peux être incluse dans le terminal utilisateur ou bien dans un serveur distant. De même, la capture du contexte peut recourir à une infrastructure extérieure, comme une grappe de satellites pour le système de localisation GPS. La figure 4 présente deux exemples de

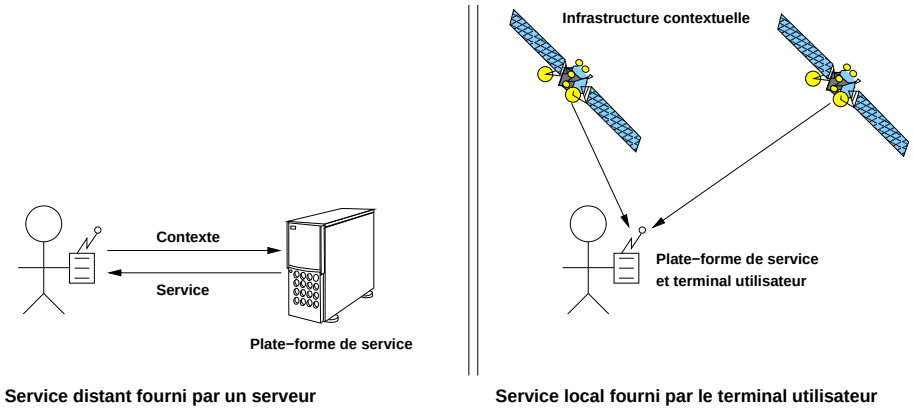

FIG. 4 – *Deux architectures possibles pour une application logique*

configurations. Le premier exploite un serveur distant pour la plate-forme de services et ne nécessite pas d'infrastructure pour la capture du contexte. Le second utilise une infrastructure contextuelle et inclut la plate-forme de services dans le terminal utilisateur.

Les applications reposants sur l'approche logique exploitent un contexte unifié ou non. Dans la suite nous présentons plusieurs applications logiques, pour chacune d'elles nous précisons donc la nature du contexte.

2.3.2 Approche physique

Contexte physique L'approche physique exploite un contexte unifié. L'espace considéré est l'espace physique et les éléments contextuels sont des données. Le contexte de l'utilisateur est donc représenté par un sous ensemble de ces données. La définition d'un contexte unifié associe aux éléments contextuels une zone contextuelle. Dans le cas présent, cette zone est représentée par une portion de l'espace physique.

Avec l'approche physique, contrairement à l'approche logique, les données ne sont plus stockées dans un serveur mais directement dans l'espace physique et occupent leur zone contextuelle. Chaque donnée est embarquée dans un calculateur sans fil. La position physique de la donnée correspond donc à celle du calculateur et sa zone contextuelle est définie par la zone de communication du calculateur : sphère pour des communications radio et cône pour des communications infrarouges. Avec un tel système, l'espace physique devient une mémoire et les données en occupent chacune une zone, de manière non exclusive. Nous considérons que la zone contextuelle représente la "forme" de la donnée : sphère, cube, cylindre ...

Nous appelons contexte physique un contexte unifié pour lequel les données sont stockées dans l'espace physique. Une application exploitant l'approche physique exploite systématiquement un contexte physique, mais elle peut aussi étendre le contexte à d'autres espaces.

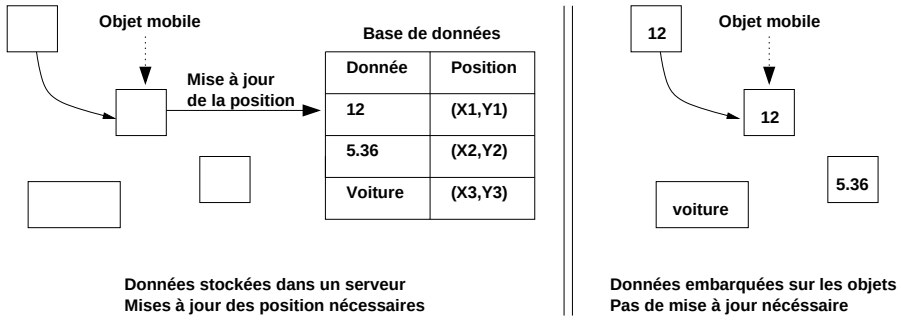

FIG. 5 – *Approche physique*

Nous présentons notamment dans la suite de document, un guide virtuel exploitant un contexte à la fois physique et Web.

Intérêt de l'approche physique Nous constatons que l'approche physique ne nécessite pas de plate-forme de services, ce qui est un réel avantage. En effet, avec l'approche logique un tel système est nécessaire pour déterminer en fonction du contexte de l'utilisateur le service à délivrer. Cette étape intermédiaire représente autant de calculs que l'approche physique n'a pas à effectuer. De plus, avec l'approche physique, les données sont stockées dans l'espace. Associer une donnée et un objet physique est donc naturel, il suffit d'embarquer le calculateur associé sur l'objet et d'y stocker la donnée. Si l'objet se déplace, elle se déplace avec lui. En revanche, avec l'approche logique associer une donnée et un objet est bien plus difficile. Il faut stocker dans la plate-forme la position des objets. Si les objets sont mobiles, des mises à jour permanentes sont nécessaires, ce qui peut rapidement surcharger la plate-forme ou le système de communication. De surcroît, un système permettant de positionner les objets devient nécessaire. La figure 5 illustre ceci, avec à gauche l'approche logique et à droite l'approche physique.

Revenons à notre exemple de guide virtuel pour musée. Avec l'approche logique, les données doivent être indexées d'après la position des œuvres associées. La plate-forme de services détermine les données à transmettre au visiteur d'après sa position. Avec l'approche physique les données sont directement embarquées sur les œuvres et occupent leur zone contextuelle. De part l'utilisation d'un contexte physique, le contexte de l'utilisateur est directement constitué des données représentant les œuvres qui l'entourent. L'application n'a donc pas besoin d'une plate-forme de services pour déterminer les informations à afficher : elles sont directement présentes dans le contexte de l'utilisateur.

La figure 6 présente cette application réalisée avec cette approche. A chaque tableau est associé un ensemble d'informations qui est ici représenté par le nom de l'artiste. Dans cet exemple, nous considérons que les données ont une forme rectangulaire englobant le tableau correspondant. Le guide du visiteur se contente d'afficher le contenu de son contexte.

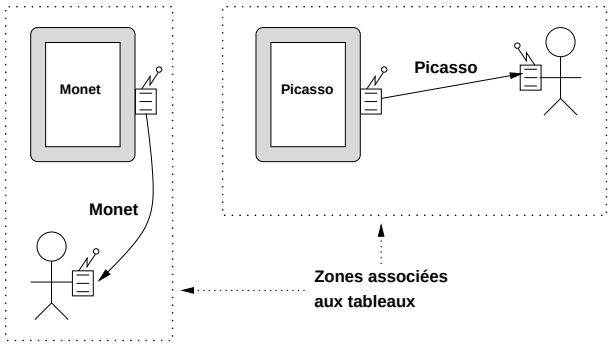

FIG. 6 – *Approche physique*

3 Applications basées sur l'approche logique

La plate-forme de services est l'élément principal d'une application programmée avec l'approche logique et la représentation du contexte détermine la manière dont elle est programmée. Par exemple, si le contexte de l'utilisateur est représenté par sa position, alors elle sera constituée d'un système permettant de stocker des données et de les mettre en relation avec des positions géographiques.

En fonction de leur mode d'exploitation du contexte, nous regroupons les applications logiques en trois grandes catégories :

- l'annotation contextuelle qui consiste à enrichir l'environnement par des notes électroniques diffusées à l'utilisateur lorsqu'il en est proche ;
- la navigation contextuelle qui situe l'utilisateur sur une carte et lui propose des services en fonction de sa position ;
- l'adaptation contextuelle qui adapte le service délivré par l'application au contexte de l'utilisateur.

Les applications peuvent ainsi être classées selon deux axes différents : le type de contexte et son mode d'exploitation. Dans cette partie nous les regroupons par type de contexte et précisons pour chacune d'elles le mode d'exploitation du contexte, sa nature (unifié ou non) et la structure de la plate-forme de services. Nous commençons par les applications exploitant la position de l'utilisateur. Nous poursuivons par celles utilisant des flux vidéo comme contexte et terminons par les applications reposant sur le marquage électronique.

3.1 Applications exploitant la localisation géographique

Avec cette première catégorie d'applications le contexte de l'utilisateur est représenté par sa position, ce qui correspond à un contexte non unifié. L'application doit donc mettre en œuvre un mécanisme de localisation. La plate-forme de services est constituée d'une

base de données contenant l'ensemble des données utilisées par l'application et d'un système permettant d'extraire les données pertinentes en fonction de la position de l'utilisateur.

Les applications exploitant la position de l'utilisateur sont principalement des applications d'annotation contextuelle et de navigation contextuelle.

3.1.1 Guide virtuels

Cyberguide [1] est un guide virtuel pour les expositions qui propose à la fois de diffuser aux visiteurs des informations contextuelles et de les situer sur une carte. Il est déployé sur des PDA et utilise un système de localisation basé sur des émetteurs infrarouge pour l'intérieur et un récepteur GPS pour l'extérieur. Cyberguide prend le parti de la simplicité en stockant toutes les données de l'application dans chaque guide. De cette manière, le terminal utilisateur fait également office de plate-forme de services et il est inutile de déployer un réseau de communication entre les guides et la plate-forme de services. Cependant, la réplication totale de toutes les informations sur chaque machine pose le problème des mises à jour lorsque les expositions sont modifiées. En effet, dans ce cas là il faut changer le contenu de chacun des guides.

Dans [34] Maccoll et al présente un système de visite partagée prenant en compte aussi bien des visiteurs physiques que virtuels. Les visiteurs virtuels visitent le musée par l'intermédiaire d'un navigateur Web ou d'un affichage 3D. Les visiteurs physiques sont dotés d'un PDA qui superpose à une vue 3D de la salle d'exposition des informations contextuelles et des personnages représentant les visiteurs virtuels. Le système de localisation est réalisé via des émetteurs à ultrasons et un compas électronique pour l'orientation. Les données contextuelles diffusées aux visiteurs sont stockées dans un serveur central qui correspond donc à la plate-forme de services pour cette application.

Ces deux applications proposent aussi bien un service d'annotation contextuelle enrichissant les objets de l'exposition, qu'un service de navigation contextuelle situant les visiteurs sur une carte.

3.1.2 Système d'Informations Géographique (SIG)

Un SIG est un système permettant la visualisation et l'exploitation de données géographiques. Ses applications sont multiples. Il est par exemple utilisé dans l'étude du trafic routier dans les villes [38], avec pour objectifs la planification urbaine, la visualisation et l'analyse des conditions de trafic. L'étude de l'impact du bruit en environnement urbain via un SIG est présentée dans [35]. Le GPS est le système de localisation utilisé dans les applications du SIG.

Bases de données spatiales Un SIG stocke ses données dans une base de données spatiales. Un tel système permet de stocker des données en leur adjoignant une forme géométrique. Les bases de données spatiales se sont développées conjointement au SIG pour lui offrir une solution de stockage des données adaptée. Une introduction détaillée aux bases de données spatiales est faite dans [22].

Un système de gestion de base de données (SGBD) classique permet de stocker et d'organiser des données selon le modèle relationnel. Une base de données spatiale fonctionne sur le même principe mais offre la possibilité d'ajouter en plus des attributs spatiaux aux objets. Les attributs spatiaux servent à modéliser le monde physique et permettent d'associer une forme à une donnée. Cette modélisation est faite à l'aide de points, de segments et de surfaces, une surface étant composée de segments et un segment de deux points. Par exemple, une rivière ou des routes sont représentées par des segments, une ville par une surface ou un point si l'échelle est plus grande. Une fois la base de données constituée, il est possible de l'interroger pour savoir, par exemple, si deux segments se coupent, ou bien si un point est à l'intérieur d'une surface.

Les bases de données d'objets mobiles [19, 51] sont une évolution des bases de données spatiales. Elles intègrent en plus une dimension temporelle dans la description des objets. Il est ainsi possible d'envoyer des requêtes demandant, par exemple, si un objet mobile est à l'intérieur d'une région à un instant t. Le SIG Arcview exploite une telle base de données.

SIG et informatique contextuelle A l'origine, les SIG n'ont pas été conçus pour l'informatique contextuelle, mais pour des applications de cartographie et d'analyse de données géographiques. Cependant, leur utilisation présente des avantages dans le cadre d'applications contextuelles nécessitant une cartographie du monde physique. Par exemple, Petchev et Claramunt [38] présentent un système permettant aux utilisateurs de recevoir sur leur téléphone mobile l'état du trafic routier dans leur secteur, avec notamment une estimation du temps de trajet. Ce système propose également un service indiquant le temps d'attente pour le prochain bus. Cette application peut être rapprochée de la navigation contextuelle, même si l'utilisateur ne visualise pas sa position sur une carte.

Le SIG est aussi employé dans les applications d'aide à la réparation de réseaux électriques [7, 26] avec pour objectif la réduction du temps de réparation. Le SIG contient une cartographie du réseau électrique et est interrogé via un réseau cellulaire. De son coté, le réparateur est équipé d'un PDA qui permet d'afficher des informations sur le réseau électrique en fonction de sa position. Il peut également visualiser sur une carte l'emplacement des installations à réparer et consulter l'historique des réparations effectuées précédemment sur l'installation défectueuse. Dans ce cas là nous sommes face à une application à la fois d'annotation et de navigation contextuelle.

Le SIG est un système logique de part l'utilisation d'un contexte non unifié, mais aussi par l'utilisation d'une base de données spatiales. Cette base de données contient une modélisation numérique du monde qui est utilisée pour délivrer le service. Elle représente le composant principal de la plate-forme de services.

3.2 Applications basées sur de la reconnaissance d'images

Nous présentons maintenant une série d'applications pour lesquelles le contexte est représenté par un ou plusieurs flux vidéo. Le contexte est capturé par un ensemble de caméras observant la scène, l'application lui applique ensuite une série de traitements pour en ex-

traire les informations utiles et délivrer le service à l'utilisateur. Ce type de contexte est non unifié.

Les programmes présentés ici sont tous deux des applications d'adaptation contextuelle. Ils délivrent un service de base qui est adapté en fonction du contexte de l'utilisateur.

3.2.1 Une approche modulaire

Crowley et al. proposent un modèle, basé sur la reconnaissance d'images, pour construire des applications contextuelles [11]. Les applications obtenues sont composées par un assemblage de modules élémentaires, à la manière de SADT [16]. Chaque module a des entrées, des sorties et des entrées de contrôle. Les auteurs les regroupent en plusieurs catégories, dont :

- les modules identifiant des propriétés physiques, comme un détecteur d'yeux dans une image;
- les modules de relations qui donnent en sortie des propriétés qui sont fonctions de celles en entrée, par exemple un module qui reçoit en entrée des positions et tailles d'yeux et donne en sortie des paires d'yeux;

Le système est illustré avec une application de travail en collaboration. Plusieurs utilisateurs sont connectés par l'intermédiaire de canaux vidéo et audio à haut débit. Du coté de chacun d'eux nous avons trois caméras : une pour observer ses yeux, une pour filmer son visage et une pour filmer le plan de travail par le dessus. De plus, chacun dispose d'un moniteur pour observer les flux vidéo émis par les autres participants. La diffusion des flux vidéo de l'utilisateur vers les autres participant représente le service de base de cette application. Cette diffusion est adaptée en fonction du contexte de l'utilisateur : s'il est en train de dessiner elle transmet la vue de la surface de travail, s'il fixe son moniteur, une vue de son visage, et s'il regarde ailleurs, une vue globale de son environnement. Avec cette application, les flux vidéo sont non seulement envoyés vers les autres participants mais également employés pour déterminer le contexte de l'utilisateur. La plate-forme de services est donc uniquement composée de la série de traitements et n'utilise pas de modèle du monde physique.

3.2.2 EasyLiving

Le projet EasyLiving [5], développé par Microsoft, a pour objectif de réaliser des environnement intelligents et s'apparente à la domotique. EasyLiving propose divers services, principalement centrés sur l'utilisation des ordinateurs situés dans la pièce. Par exemple, lorsque l'utilisateur se trouve sur le canapé son environnement est affiché sur le mur avec un vidéo projecteur. Lorsqu'il se déplace vers un couple moniteur/clavier pour s'isoler et consulter son courrier, l'affichage y est automatiquement déplacé. Ici, le service de base est l'utilisation d'un ordinateur. L'adaptation contextuelle du service consiste en la sélection automatique du poste de travail en fonction du contexte de l'utilisateur.

Le contexte de l'utilisateur est capturé par une série de caméra filmant la scène. En plus des flux vidéo, EasyLiving exploite une modélisation géométrique de l'environnement pour déterminer les services à délivrer. Cette modélisation décrit l'agencement de la pièce

et notamment la position des dispositifs informatiques. L'architecture d'EasyLiving est typiquement logique, la plate-forme de services reçoit en entrée les flux vidéo et s'appuie sur le modèle géométrique pour fournir les services.

3.3 Applications basées sur le marquage électronique

Le principe du marquage électronique consiste à ajouter aux objets de la vie courante des étiquettes détectables à distance. Plusieurs technologies sont disponibles, nous pouvons citer les étiquettes radio (RFID) [27, 32, 49], les codes barre [13, 42, 43] ou même les liens HTML [30,31]. Nous présentons ci-après ces différentes solutions ainsi qu'un ensemble d'applications.

Les applications présentées dans cette partie exploitent la détection d'étiquettes pour délivrer leur service. Les étiquettes sont détectées par un lecteur d'étiquettes qui a une portée de détection dépendant de la technologie employée. L'ensemble des étiquettes détectées par le lecteur représente son contexte. Si ce lecteur est porté par un utilisateur, l'ensemble correspond également à son contexte. D'après la définition (3) ceci correspond à un contexte unifié. L'espace considéré est l'espace physique et les éléments contextuels sont les objets détectés via les étiquettes. Nous avons la zone contextuelle de E ($S(E)$) qui correspond à la zone de détection du lecteur d'étiquette.

Bien que le contexte exploité soit unifié, ces applications restent des applications logiques car les données n'occupent pas directement leur zone contextuelle, mais sont stockées dans un serveur distant. Pour fournir des services, les applications doivent établir une correspondance entre les étiquettes détectées et les actions à effectuer, ce qui est réalisé via une base de données associant étiquettes et données. La machine chargée de cette tâche est la plate-forme de services. Elle reçoit en entrée les étiquettes détectées par le lecteur, effectue une correspondance entre étiquettes et données et délivre finalement le service.

3.3.1 RFID

Principes de base Les étiquettes RFID sont conçues pour être ajoutées aux objets de la vie courante, tout comme les classiques codes barre. Leur lecture est faite par ondes radio et donc, contrairement aux codes barre, une ligne de vue n'est pas nécessaire pour les détecter. Ainsi, ils peuvent être rendus invisibles en les intégrant directement dans les objets. La distance maximum de lecture varie de quelques centimètres à plusieurs mètres selon le type d'étiquette. Les RFID sont soit passifs soit actifs. Les premiers contiennent juste un numéro unique et n'ont pas besoin d'alimentation, tandis que les seconds contiennent une petite zone de mémoire accessible par le lecteur et requièrent une alimentation.

Applications Dans [44] les auteurs présentent un projet de supermarché électronique proposant de nouveaux services à ses clients. Ainsi, le prix du caddie est calculé en temps réel : le client n'a donc plus besoin de le vider lors de son passage en caisse. Il se voit également proposer des offres promotionnelles en relation avec les produits qu'il est en train d'acheter. Par exemple, lorsqu'il vient de mettre un produit du rayon informatique dans son caddy, son terminal lui diffuse les promotions faites dans ce rayon susceptibles de l'intéresser.

Ce type d'application propose également des services en dehors du supermarché. Lorsque le client est chez lui, les appareils équipés de lecteur RFID consultent les produits achetés pour établir une liste de commissions contenant les articles les plus courants. Cette liste peut être utilisée, la fois suivante, pour consulter les supermarchés environnants et trouver la meilleure offre. Cette application est déployée en embarquant un RFID sur l'ensemble des produits, et implémentée via une base de données faisant la correspondance entre articles et RFIDs. De son coté, le client dispose d'un caddy équipé d'un lecteur de RFID et d'un PDA pour la diffusion des informations. Un supermarché électronique est typiquement une application d'annotation contextuelle. Ici les produits sont enrichis par des informations numériques telles que leur prix ou bien leurs ingrédients.

Römer et Schoch proposent une architecture permettant de bâtir des applications sur la notion d'événements [27]. Les objets physiques embarquent tous un RFID. L'apparition et la disparition d'un RFID représentent les événements de base auxquels peuvent s'abonner les composants de l'application. Ce système est utilisé pour réaliser un assistant cuisinier virtuel [27] qui propose des recettes en fonction des aliments posés sur le plan de travail. Cette application est réalisée en ajoutant sur les emballages des aliments une étiquette électronique et en équipant le plan de travail d'un lecteur. Les événements surveillés sont l'apparition et la disparition d'un aliment. Chaque fois que l'un d'eux se produit, la liste des recettes possibles est mise à jour. Cette application appartient à la catégorie des applications d'adaptation contextuelle, le service de base étant la proposition de recettes de cuisine.

Les travaux de Want et al visent quant à eux à enrichir des documents au moyen de RFID [49]. Pour cela, ils adjoignent des étiquettes à des livres, des imprimantes ou même des cartes de visite. Plusieurs applications sont proposées. Par exemple : si l'on présente un livre devant un lecteur de RFID la page Web de l'auteur est affichée, ou bien lorsque le lecteur détecte une carte de visite, un mail vide destiné à l'émetteur de la carte est créé. De par l'enrichissement des documents, ce type d'application se rapproche des applications d'annotation contextuelle.

3.3.2 Codes-barres

Les codes-barres sont une autre solution pour réaliser des étiquettes électroniques. Rekimoto et al ont développé deux systèmes. Le plus ancien [43] propose des codes barre réalisés sous la forme d'une succession de bandes verticales rouges et bleues. Dans la seconde approche [42] les codes-barres sont représentés par une grille de carrés noirs et blancs et baptisé CyberCodes. López, quant à lui, propose des codes-barres circulaires [13] où l'information est encodée avec des secteurs d'anneaux noirs et blanc.

Toutes ces technologies reposent sur l'utilisation d'une caméra et de techniques de traitements d'images pour la reconnaissance des étiquettes. En plus de détecter la référence de l'étiquette, ces traitements permettent également d'obtenir la position relative de l'étiquette par rapport à la caméra, aussi bien avec les codes-barres circulaires que les CyberCodes. Notons que toutes ces techniques sont consommatrices de puissance de calcul et impliquent l'utilisation d'une machine puissante pour réaliser les traitements.

Les codes-barres présentent certains avantages par rapport aux RFID. Leur coût de fabrication est négligeable, une simple imprimante suffit. De plus, ils permettent d'obtenir la position relative de l'étiquette par rapport à la caméra. Leur inconvénient majeur est la nécessité d'une ligne de vue pour effectuer la lecture. Ceci implique que les codes-barres doivent être visibles à la surface des objets et les rend inadaptés pour les situations où l'esthétique prime. Les RFID peuvent être rendus invisibles. En revanche, leur coût est supérieur et il n'est pas possible d'avoir la position de l'étiquette ; l'application sait seulement que l'étiquette est à portée du lecteur.

Les applications exploitant les codes-barres sont du même type que celles utilisant les RFID. López a développé un système, rentrant dans la catégorie de l'adaptation contextuelle, qui déplace l'application de l'utilisateur sur l'ordinateur le plus proche. Rekimoto propose d'utiliser son système pour des applications d'annotation contextuelle et notamment pour les guides de musées. Les codes-barres, disposés à coté des œuvres, sont lus par le guide qui diffuse ensuite l'information associée. Il propose également d'utiliser les codes-barres dans le cadre de la réalité enrichie pour associer les étiquettes à des objets tridimensionnels. Ces objets sont ensuite surimposés dans l'environnement, là où se trouve l'étiquette.

3.3.3 Étiquettes Web

L'objectif du projet CoolTown [30, 31] est de fournir une "présence" sur le Web aux personnes, lieux et objets physiques. Une page Web est donc associée à chaque entité et son adresse est accessible dans son voisinage. Les techniques permettant de découvrir les adresses (URL) sont multiples. La détection peut être directe, via un faisceau infrarouge, ou indirecte en détectant une étiquette RFID et en faisant ensuite appel à un serveur qui renverra l'URL correspondante. Dans ces cas là, les URL ne sont détectées qu'a proximité des objets physiques. L'application peut également faire appel à un service de découverte global qui renvoie l'adresse des objets délivrant le service recherché. Une fois l'adresse d'un objet détectée, le terminal l'utilise pour récupérer et afficher la page Web correspondante. L'argument en faveur de l'utilisation de la technologie Web est sa simplicité et son fort niveau de déploiement actuel.

Les applications proposées par CoolTown sont multiples et similaires à celles des systèmes présentés jusqu'alors. Les applications d'annotation sont particulièrement visées via l'enrichissement d'objets avec des informations numériques. CoolTown permet également d'accéder aux services proposés par un objet via sa page Web, par exemple un service d'impression pour une imprimante.

Avec CoolTown la plate-forme de services correspond au serveur Web qui délivre le service. Elle reçoit en entrée des requêtes basées sur le protocole Web (http) et décrivent le service requis. Le contexte est représenté par les URL détectées autour de lui.

3.4 Synthèse

Dans cette partie nous avons présenté plusieurs applications et systèmes logiques en les regroupant selon trois grandes catégories. Pour chacun d'eux nous avons notamment

explicité le type d'application, la composition de la plate-forme de services ainsi que le modèle éventuel. Pour la première catégorie d'applications le contexte de l'utilisateur est représenté par sa position. La plate-forme de services utilise cette position pour fournir un service en s'appuyant sur une base de données stockant l'ensemble des données contextuelles. La seconde catégorie d'applications exploite des flux vidéo en tant que contexte de l'utilisateur. La plate-forme applique une série de traitements à ces flux pour déterminer le service. Enfin, la troisième catégorie d'application représente le contexte de l'utilisateur par un ensemble d'étiquettes. La délivrance du service a lieu en effectuant une correspondance entre étiquettes et données.

Les applications qui exploitent la reconnaissance d'images et celles qui emploient la localisation géographique utilisent un contexte non unifié. Seules celles basées sur le marquage électronique utilisent un contexte unifié. Si nous considérons les applications exploitant la localisation géographique nous remarquons que la position n'est en fait utilisée que pour obtenir le contexte unifié de l'utilisateur. La plate-forme de services détermine à partir de la position de l'utilisateur les objets qui l'entourent, pour en déduire ensuite les données correspondantes. Le cas des applications reposant sur l'analyse de flux vidéo est différent. D'après la définition de Dey et Abowd le contexte pour de telles applications est représenté par les flux vidéo. Cependant, si l'on observe le comportement de ces applications, nous remarquons qu'avant de délivrer le service elles cherchent tout d'abord à extraire du contexte des informations plus précises sur la situation de l'utilisateur. Le contexte brut n'est pas directement utilisable. Par exemple, EasyLiving recherche la position de l'utilisateur. L'application de Crowley et al, quant à elle, détecte l'orientation du regard de l'utilisateur.

Lorsque le contexte n'est pas unifié, les applications doivent lui faire subir une série de transformations afin d'obtenir les informations nécessaires à la délivrance du service. En revanche, pour les applications basées sur le marquage électronique, le contexte est unifié et directement exploitable. Ceci nous montre bien l'importance d'une définition précise du contexte lors de la phase de conception d'une application ; un contexte bien défini simplifie d'autant plus l'architecture de l'application.

4 Systèmes pour construire des applications physiques

L'approche physique est caractérisée par l'exploitation d'un contexte physique. L'espace physique est donc peuplé de données, chacune occupant sa zone contextuelle. Une donnée fait partie du contexte de l'utilisateur lorsque celui-ci se trouve à l'intérieur de la zone contextuelle correspondante.

Cette approche pose le problème de l'accès aux données. En effet, il est possible qu'un utilisateur soit à l'intérieur de plusieurs zones simultanément. Ainsi, son contexte physique peut être composé de plusieurs données. Par exemple, sur la figure 7 nous avons un utilisateur mobile et plusieurs données disposées dans l'espace, chacune occupant une zone précise : sphère, cube ... Selon sa position sur sa trajectoire, son contexte physique contient une ou plusieurs données. Les applications physiques exploitent les données présentes dans le contexte physique, elles doivent donc définir une méthode de sélection des données.

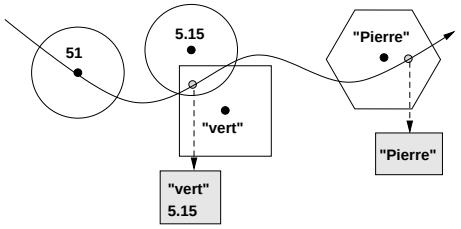

FIG. 7 – *Contexte physique d'un utilisateur mobile. Les rectangles gris représentent le contexte en deux endroits de la trajectoire de l'utilisateur.*

Les applications physiques sont principalement des applications d'annotation contextuelle et de partage de données entre utilisateurs. Avec une application de partage de données, les utilisateurs mettent à disposition d'autres personnes des données personnelles. Un individu peut accéder aux données d'un autre lorsque celui-ci fait partie de son contexte.

Les applications physiques utilisent toutes le même type de contexte, cette partie est donc organisée différemment de la précédente. Nous ne regroupons pas les applications par type de contexte, mais présentons séparément les différents systèmes existant pour réaliser des applications avec l'approche physique. Nous détaillons notamment leur mode de sélection des données, leur modèle de programmation, ainsi que les applications qu'ils permettent de réaliser.

4.1 Systèmes reposant sur un espace de tuples

4.1.1 Rappels sur Linda

Principes de base A l'origine, Linda est un langage pour systèmes distribués qui permet aux nœuds de communiquer par l'intermédiaire d'une mémoire partagée appelée espace de tuples. Le modèle de programmation Linda [20] permet de résoudre simplement le problème l'accès aux données dans le cadre d'un système physique. En effet, Linda propose un mécanisme permettant de sélectionner une donnée parmi plusieurs en se basant sur leur signature.

Linda défini un tuple comme étant une suite ordonnée d'éléments typés : entiers, flottants et chaînes de caractères. Par exemple : `<10,"Voiture",12.5>`. Un motif est un tuple dans lequel zéro ou plusieurs champs n'ont que leur type de défini. Par exemple : `<10, string, 12.5>`. Il y a correspondance entre un tuple et un motif lorsque : ils ont le même nombre d'éléments, les types de chaque élément se correspondent deux à deux et les valeurs des champs sont égales deux à deux. Ainsi, le tuple `<12.5,"voiture">` correspond au motif `<12.5,string>`. Un motif défini un ensemble de tuples qui lui correspondent.

Linda stocke les tuples dans un espace de tuples et défini plusieurs opérations pour les manipuler :

– `out(t)` ajoute le tuple `t` ;

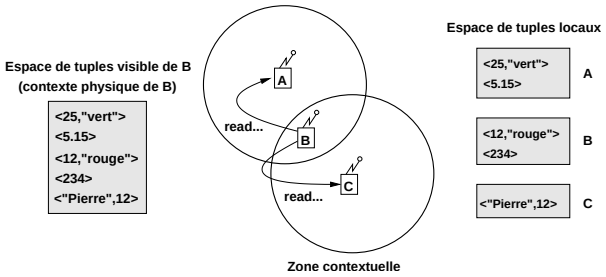

FIG. 8 – *Un espace de tuples distribué*

– `in(m)` renvoie le tuple `t` qui correspond au motif `m` et le retire de l'espace de tuples ;
– `read(m)` est identique à `in` hormis que le tuple renvoyé n'est pas retiré.

Si plusieurs tuples correspondent à un motif, Linda choisi au hasard le tuple renvoyé par `in` ou `read`. Les opérations `in` et `read` restent bloquées jusqu'à l'obtention d'un tuple, ce qui permet de synchroniser les processus sur l'occurrence d'un tuple.

Intérêt pour l'approche physique Le contexte physique de l'utilisateur est représenté par l'ensemble des données qui lui sont accessibles. Si nous choisissons de représenter ces données par des tuples, alors le mécanisme de correspondance entre tuples et motifs permet de sélectionner simplement une donnée depuis le contexte.

Du point vue de l'implantation, ceci est réalisé en ajoutant dans chaque calculateur un espace de tuples. La zone contextuelle des tuples correspond à la zone de communication du calculateur. Le contexte de l'utilisateur est alors représenté par l'ensemble des tuples accessibles. Cet ensemble correspond à l'union des espaces de tuples des calculateurs avec lesquels il peut communiquer. Lorsque l'application cherche à extraire une donnée particulière de son contexte, le système envoie une requête à chacun des nœuds connectés. Nous avons donc un espace de tuples distribué sur plusieurs machines, mais qui est vu comme un espace de tuples global par les applications. La figure 8 illustre ceci. Nous avons trois calculateurs, chacun associé à une zone contextuelle. Dans un souci de lisibilité nous ne représentons pas la zone du calculateur B. L'espace visible du calculateur B (contexte physique) est constitué de l'union des espaces de tuples locaux des trois calculateurs. Lorsque B cherche une donnée dans son contexte, en plus d'interroger son espace local une requête est envoyée vers A et une autres vers C.

De plus, les mécanismes de synchronisation de Linda, permettent de synchroniser l'application sur l'apparition d'un contexte particulier. Lorsque l'application cherche une donnée particulière par l'intermédiaire d'un `in`, l'opération restera bloquée jusqu'à l'apparition du tuple correspondant dans le contexte et donc jusqu'à ce que le contexte physique de l'utilisateur concorde. Détecter un contexte revient donc à détecter les tuples correspondant.

4.1.2 Lime

Principes de base LIME [39] est un middleware développé pour aider à la programmation d'applications distribuées incluant des nœuds fixes et mobiles. Il est fondé sur la notion d'agents mobiles et d'espace de tuples partagé. De notre coté, nous nous intéressons au cas des nœuds mobiles et considérons qu'ils sont équipés d'une interface réseau sans fil.

Un agent mobile est un programme capable d'arrêter son exécution, de migrer sur un autre nœud via le réseau et finalement de reprendre son exécution. Il est autonome dans le sens où la décision de migrer lui appartient. Les raisons d'une migration sont diverses, cependant, la plupart du temps un agent migre parce qu'il a besoin d'une ressource ou de données présentes sur un autre nœud. Par exemple, s'il doit effectuer un traitement sur un ensemble de données distantes, il migrera sur le nœud concerné pour l'effectuer localement. Contrairement à l'approche client/serveur, en migrant directement sur les nœuds pour effectuer leurs traitements, les agents mobiles permettent d'économiser de la bande passante et supportent plus facilement des déconnexions réseau. LIME qualifie de logique la migration d'un agent , au sens ou l'agent s'est déplacé sur une autre unité d'exécution. La mobilité des nœuds correspond elle à une mobilité physique.

Pour leurs communications, Lime fournit aux agents un mécanisme d'espace de tuples distribué. Ainsi, chacun d'eux possède un espace de tuples qui se déplace avec lui, l'espace de tuples visible est constitué de l'union des espaces de tuples des agents connectés. Un agent est considéré comme connecté lorsque son nœud hôte l'est. Les auteurs ont choisit le modèle Linda pour ses possibilités de synchronisation.

L'architecture de LIME permet sans conteste de créer des contextes physiques et donc de programmer des applications contextuelles avec l'approche physique. En effet, un agent ne peut accéder à l'espace de tuples d'un autre que s'il lui est connecté, autrement dit que s'il est dans son périmètre de communication. Ainsi, même si ce n'est pas défini explicitement dans le modèle de conception de LIME, nous pouvons considérer que les tuples sont stockés dans l'espace et ont pour zone contextuelle le périmètre de communication leur hôte. Dans le cas d'interface réseau utilisant des ondes radio, les tuples ont tous une forme sphérique. Seule la version mobile de Lime peut être considérée comme physique, dans le cas d'un réseau filaire les tuples ne sont plus locaux, mais accessibles depuis n'importe quel point. Ils ne peuvent donc plus être considérés comme étant des données contextuelles.

Modèle de programmation Les agents accèdent à l'espace de tuples avec des opérations similaires à celles définies par Linda, la différence étant qu'un agent peut spécifier si l'opération agit sur son espace de tuples ou bien sur celui d'un autre. Ainsi, un agent A peut ajouter un tuple en précisant qu'il sera stocké dans l'espace de tuples d'un agent B. Si B n'est pas connecté à A, le tuple est stocké dans l'espace de tuples de A jusqu'à la connexion de B. Un agent peut également exécuter une requête de type `read` en précisant si la recherche est faite sur l'espace de tuples global ou seulement sur l'espace d'un agent donné.

Aux primitives classiques de Linda LIME rajoute un mécanisme de réactions. L'opération `reactsTo(s,m)` exécute le code `s` lorsqu'un tuple correspondant au motif `m` est ajouté. LIME surveille par l'intermédiaire d'un processus séparé l'ensemble des réactions pendantes. Une

réaction est déclenchée lors de l'apparition d'un tuple correspondant, qui fait suite soit à l'insertion du tuple par l'agent lui-même, soit à la connexion d'un agent possédant le tuple correspondant. Autrement dit, la mobilité physique des agents (connexions et déconnexions) peut déclencher des réactions. Notons que Linda offre aussi des possibilités de programmation réactive. Cependant, si l'on veut surveiller l'occurrence de plusieurs tuples simultanément, il nous faut un processus par tuple surveillé. Cette solution n'est pas performante et peu pratique.

Applications Parmi les applications réalisables avec LIME, les auteurs présentent notamment un jeu consistant en une chasse au drapeau faite par deux équipes [40]. L'application diffuse en permanence une vue de l'environnement incluant les objets proches ainsi que la position des coéquipiers. L'intérêt de ce jeu réside dans les similarités qu'il partage avec des situations réelles telles que l'exploration d'un terrain inconnu par un groupe de robot.

4.1.3 Spread

Principe de base SPREAD [8] est un environnement dédié à la programmation d'applications contextuelles avec l'approche physique. Il propose une architecture permettant de réaliser et manipuler aisément des contextes physiques. SPREAD part de la constatation qu'une donnée contextuelle est en général associée à une entité physique : lieu, objet ou personne. Les calculateurs sans fil gérant les données contextuelles sont donc embarqués sur les entités correspondantes. Contrairement à LIME, SPREAD spécifie explicitement dans son modèle que chaque donnée occupe un volume précis, définissable à volonté : cube, sphère, cylindre ... Cependant, la zone étant définie par la portée de communication du calculateur associé, les technologies de communication employées actuellement limitent la forme à une sphère. SPREAD étant un système physique, il doit fournir un mécanisme à l'application pour extraire une donnée particulière du contexte. Tout comme LIME, SPREAD fait le choix d'utiliser un espace de tuples.

Modèle de programmation SPREAD utilise un mécanisme d'espace de tuples distribué. Chaque nœud est équipé d'un espace de tuples et dispose de plusieurs opérations pour accéder à l'espace de tuples : `read` qui recherche un tuple, `out` qui ajoute un tuple, `capture` qui renvoie un ensemble de tuples sans les supprimer, `check` qui teste la présence d'un tuple et `drop` qui supprime un tuple local. Les opérations `read`, `capture` et `check` agissent sur l'espace de tuples global. `drop` et `out` sont des opérations locales. Le modèle Linda a été choisi parce qu'il permet de résoudre le problème de la sélection des données, mais aussi pour son mécanisme de synchronisation et la simplicité des programmes réalisés.

Applications SPREAD permet de développer des applications contextuelles exploitant le contexte physique de l'utilisateur. Un guide pour exposition a notamment été réalisé et expérimenté à la *Cité des Sciences* [9]. Cette application partage avec CoolTown l'utilisation d'étiquettes Web pour enrichir l'environnement. Les objets exposés sont donc associés à

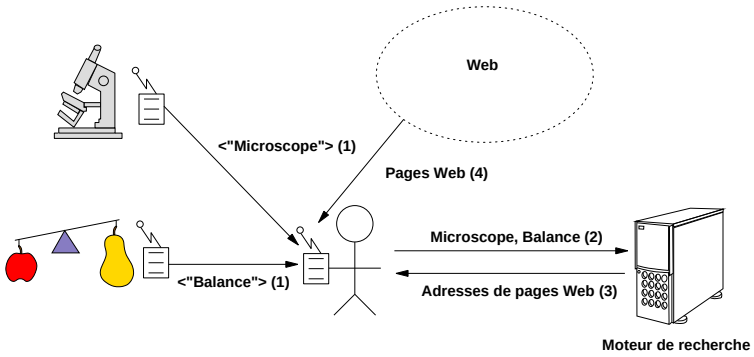

FIG. 9 – *Un guide virtuel exploitant un contexte unifié*

des tuples représentant soit un mot le décrivant soit l'adresse d'une page Web (URL) le concernant.

Le contexte de l'utilisateur est constitué selon deux espaces distincts, l'espace physique et le Web. Les mots et URL représentent le contexte physique. Les mots sont utilisés pour effectuer une requête avec vers un moteur de recherche qui renvoie l'URL des pages Web les plus proches. Ces pages représentent le contexte Web de l'utilisateur. Le moteur de recherche évalue la distance entre une page Web et un groupe de mots en regardant, par exemple, si les mots sont présents dans la page ou bien s'ils apparaissent dans son titre. Finalement, le guide propose à l'utilisateur aussi bien les URL présentes dans son contexte physique que son contexte Web, afin de lui fournir des informations sur les objets ou expériences qu'il regarde.

La figure 9 présente cette application. Les mots sont distribués sur les stands d'exposition via un calculateur sans fil qui peut communiquer avec le guide du visiteur. Le guide détecte tout d'abord les mots, puis fait une requête vers le moteur de recherche et utilise le résultat pour charger les pages Web correspondantes. Par souci de clarté, la figure ne représente pas les URL directement associées aux objets exposés, ni les zones contextuelles.

4.2 PeerWare

4.2.1 Partage des données

PeerWare [12] est un middleware dédié à la création d'applications mobiles et/ou pair à pair. PeerWare est proche de LIME mais ne repose pas sur un espace de tuples. Il est basé sur une structure de données appelée GVDS pour : Global Virtual Data Structures. Le principe de partage des données dans un GVDS est similaire à celui utilisé dans un espace de tuples distribué : les calculateurs connectés ont la vision d'un espace commun correspondant à l'union des espaces locaux. Un GVDS ne définit pas la structure des données ni les opérations pour y accéder, c'est à l'application de le faire. PeerWare fait le choix de

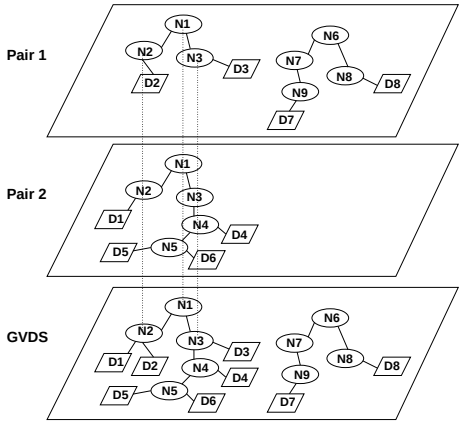

FIG. 10 – *Partage des informations dans PeerWare*

représenter les données par une forêt d'arbres ayant des racines distinctes. Les feuilles des arbres sont des documents et correspondent aux données partagées par les nœuds. Lorsque plusieurs pairs sont connectés les nœuds qui portent le même nom et occupent la même place dans l'arbre sont représentés par un seul nœud dans la structure globale (cf figure 10). Avec une telle structure, la sélection d'une donnée ne se fait plus par l'intermédiaire de sa signature comme dans Linda, mais via sa place dans l'arborescence. Un programme peut par exemple modifier l'ensemble des documents attachés à un nœud, la position du nœud étant passée en paramètre. Cette organisation est similaire à l'organisation d'un système de fichiers, à la différence près qu'un système de fichiers a une racine unique.

A partir du moment où PeerWare est déployé sur des terminaux mobiles, il permet, pour les mêmes raisons que LIME, de réaliser et de manipuler des contextes physiques. Il est donc adapté au développement d'applications contextuelles reposant sur l'approche physique.

4.2.2 Modèle de programmation

PeerWare propose deux catégories d'opérations sur les données : les opérations agissant sur les données locales et celles sur les données globales. Toutes les opérations de création, modification et effacement de nœuds ou documents sont locales. La dernière opération locale correspond au signalement d'un événement. PeerWare propose trois opérations opérant sur la structure globale : `execute`, `subscribe` et `executeAndSubscribe`. La première opération exécute une action sur un ensemble de documents. Elle prend en paramètre un ensemble de filtres qui déterminent les documents sur lesquels l'action agira. Cette opération permet de disséminer sur les calculateurs des opérations à effectuer localement. La seconde opération permet à un programme de s'enregistrer pour l'occurrence d'un événement. Elle prend également un ensemble de filtres en paramètre qui déterminent les documents sur

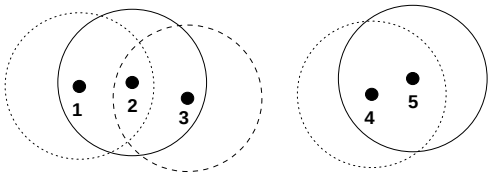

FIG. 11 – *Exemple de formation de SIS. Un calculateur appartient à un SIS s'il peut communiquer avec le nœud de référence.*

lesquels l'événement surveillé doit arriver. La dernière opération est une combinaison des deux premières.

4.2.3 Applications

Une des applications envisagées par PeerWare est l'aide aux sauveteurs en cas de catastrophe naturelle. Dans ce cas là l'application ne peut se reposer sur une infrastructure globale. Le scénario d'un ensemble de nœuds mobiles communicant en mode pair à pair prend alors tout son sens. PeerWare permet aux sauveteurs de partager des informations concernant le lieu d'opération ainsi que d'envoyer et recevoir des notifications à propos d'événements appropriés.

4.3 Système d'informations spontanés (SIS)

4.3.1 Principe

L'idée de système d'informations spontané [2] est issue du mode d'établissement d'une conversation entre plusieurs individus : ils ne peuvent communiquer que s'ils sont suffisamment proches pour s'entendre. Ainsi, si plusieurs personnes engagent une conversation et si chacune d'elles porte un calculateur sans fil, dans le même temps ceux-ci pourront également rentrer en communication. L'ensemble des calculateurs voisins avec lesquels un nœud peut communiquer est appelé SIS.

Considérons un ensemble de calculateurs (nœuds) mobiles équipés d'une interface réseau sans fil. Lorsque deux calculateurs ou plus sont suffisamment proches pour pouvoir communiquer, il y a alors formation de SIS au sein desquels les nœuds peuvent communiquer et échanger des données. Le SIS associé à un calculateur disparaît lorsqu'il n'a plus de voisin. La figure 11 présente cinq calculateurs mobiles, chacun étant associé à un SIS. L'ensemble des calculateurs composant un SIS est dynamique, il change au fur et à mesure de leurs mouvements. Le modèle SIS est un support pour les applications contextuelles et notamment les réseaux de personnes. Nous en présentons plusieurs à la fin de cette partie.

Nous fournissons maintenant une définition plus formelle d'un SIS issue de [2]. Un nœud mobile est noté N_i et sa zone de communication $ZC(N_i)$. Lorsqu'un nœud N_j est à l'intérieur de la zone de communication de N_i, nous notons $N_j \subset ZC(N_i)$. Le prédicat CV définit la condition nécessaire pour avoir une communication par voisinage entre deux nœuds N_i et N_j à un instant t_i (temps local à N_i) :

$$CV(N_i,N_j,t_i) \Leftrightarrow \exists t_i / N_i \subset ZC(N_j) \wedge N_j \subset ZC(N_i)$$

Nous notons $SIS(N_i)$ le SIS associé au nœud N_i. Nous avons finalement :

$$SIS(N_i) = \{N_j / \forall j, \exists t_i / CV(N_i,N_j,t_i)\}$$

Les systèmes d'informations spontanés sont des systèmes typiquement physiques. De même que pour les système précédents, en raison de la portée limitée des calculateurs, les SIS permettent de réaliser des contextes physiques. Ainsi, les SIS sont adaptés au développement d'applications contextuelles avec l'approche physique. Le contexte de l'utilisateur est représenté par l'ensemble des données accessibles. Tout comme PeerWare, le modèle SIS est un modèle de communication entre calculateurs mobiles, il ne spécifie pas le mode d'accès aux données ni leur représentation. C'est à l'application de s'acquitter de ces tâches.

4.3.2 Applications

SIDE Surfer est un système permettant des échanges de données, en mode pair à pair, basés sur les centres d'intérêts des utilisateurs. Lorsqu'ils sont proches les uns des autres, lors d'une réunion par exemple, leur PDA forment une série de SIS afin d'échanger des documents. Pour cela, chaque utilisateur stocke ses données personnelles sur son PDA, en les indexant avec des mots clef. Des techniques de data mining sont ensuite appliquées pour créer un profil utilisateur à partir de ces mots clef. Lorsque plusieurs utilisateurs se rencontrent, leurs PDA échangent spontanément les documents en se basant sur les profils respectifs.

PERSEND, qui fait suite à SIDE surfer, a pour objectif la prise en compte de la volatilité des données et de leur aspect dynamique dans les applications contextuelles. Avec un système de base de données classique, le résultat d'une requête est calculé d'après le contenu de la base au moment de la requête. Si l'on veut suivre l'évolution des données de manière continue, ce système impose de faire des requêtes régulièrement. Les requêtes continues proposent une solution élégante à ce type de problèmes. Une requête continue n'est pas exécutée une fois pour toute par la base de données, mais reste active après le calcul du résultat pour être réévaluée à chaque ajout ou suppression d'enregistrements. Ainsi, le résultat obtenu évolue dynamiquement au fur et à mesure des opérations sur la base. PERSEND propose de prendre en compte l'aspect dynamique des données au sein d'un SIS via un mécanisme de requêtes continues. Chaque calculateur embarque une base de données et peut lancer une requête (continue ou classique) aussi bien sur la base de données locale que sur celle des nœuds de son SIS. Le résultat d'une requête évolue donc en fonction des opérations effectuées sur les bases de données et également selon les départs et arrivées de nœuds, un départ ou une

arrivée se traduisant par l'ajout ou la suppression des enregistrements provenant du nœud correspondant.

De part leur mobilité et leur portée de communication réduite, la durée de connexion entre les nœuds d'un SIS est variable. Dans [48], les auteurs présentent des schémas prédictifs évaluant la durée de la fenêtre de connexion et permettant aux applications de l'exploiter de manière optimum. Ces schémas reposent sur l'utilisation de la durée de connexion, du comportement de l'application et d'une politique d'ordonnancement pour l'envoi des paquets. Les résultats montrent une meilleure exploitation de la bande passante qu'avec la méthode de communication classique, tout en transmettant autant de paquets.

4.4 Réseaux de capteurs

4.4.1 Présentation

Un réseau de capteurs est un ensemble de dispositifs miniatures capables de communiquer avec leurs voisins et de mesurer une ou plusieurs grandeurs physiques. L'objectif est d'instrumenter l'environnement. Les applications envisagées sont multiples. On peut par exemple projeter de larguer au dessus d'un incendie une myriade de capteurs, ceux qui tomberont dans le feux seront détruis, les autres fourniront une cartographie de l'incendie facilitant le travail des pompiers.

Les applications des réseaux de capteurs vont au-delà de la mesure de grandeurs physiques. En effet, les architectures matérielles employées ne limitent pas ces appareils à une simple tâche de détection. Ils sont aussi capables d'effectuer des calculs ainsi que de stocker des données dans leur mémoire. Ils doivent être considérés comme des calculateurs sans fil sensibles à leur environnement physique. Ils peuvent par exemple être embarqués sur des objets physiques afin de réaliser une application de gestion et suivi de stock.

L'architecture globale d'un réseau de capteurs est constituée de trois couches [46] : l'infrastructure matérielle, les protocoles réseaux et les applications. Actuellement, l'essentiel des travaux réalisés se concentrent sur la couche protocole. Nous pouvons notamment citer les protocoles de localisation [17, 33] ainsi que les protocoles de diffusion directe [3, 18]. A notre connaissance, aucune application contextuelle n'a encore été développée spécifiquement pour les réseaux de capteurs. Selon nous, cette architecture présente des avantages pour le déploiement d'applications physiques. Nous les présentons dans la partie suivante, après avoir détaillé quelques exemples d'architecture matérielle.

4.4.2 Architectures disponibles

Plusieurs architectures de capteurs sont actuellement développées. Les principales sont : Smart Dust [28], les Motes [24] , les Smart Its [25] et PicoRadio [41]. Ces architectures présentent des similitudes : faible encombrement, puissance de calcul et mémoire réduite ... Nous présentons maintenant deux d'entre elles : Smart Dust parce qu'il nous semble être le projet le plus ambitieux et les Motes pour leur maturité. Les Smart Its proposent une ar-

chitecture proche des Motes. PicoRadio a pour objectif de fournir des capteurs consommant un minimum d'énergie et se concentre sur l'aspect communication.

Smart Dust Le projet Smart Dust [28] a pour objectif de fabriquer des réseaux de capteurs dans lesquels les nœuds auront une taille de l'ordre du grain de poussière. Les auteurs estiment possible de réaliser des nœuds suffisamment légers pour pouvoir rester suspendus dans l'air. Actuellement, la taille correspond à un cube de deux à trois millimètres de coté. Les capteurs incluent une unité de calcul, une interface communication laser, une unité de traitement du signal, une batterie et des capteurs solaires. Les nœuds peuvent communiquer soit entre eux soit avec une station de base.

Smart Dust est un projet très prometteur et présente des avantages indéniables pour le déploiement de systèmes spatiaux. Le coût de fabrication de ce type de capteur devrait être suffisamment faible pour embarquer un ou plusieurs nœuds sur l'ensemble des objets de la vie courante ou bien diffuser et capter de l'information depuis l'espace physique.

Motes Les Motes [24] ont été développées parallèlement à Smart Dust. Ce sont des nœuds embarquant une unité de calcul, de la mémoire, une interface de communication radio ainsi que des capteurs adaptés à la mesure de différentes grandeurs physiques : température, luminosité, humidité ... Le système d'exploitation employé est TinyOS. Il est basé sur une structure à événements et a été développé tout spécialement pour les Motes. Plusieurs versions de Motes ont été développées. Les premières ont une taille proche de celle d'une pièce de monnaie, les secondes sont plus encombrantes et ont des dimensions de quelques centimètres.

Les Motes sont plus encombrantes que les nœuds Smart Dust, mais présentent l'avantage d'être dors et déjà commercialisées. Elles ont notamment été utilisées dans [21] pour des expérimentations sur la localisation de capteurs. Notons que dans [29] une architecture partageant les mêmes objectifs que les Motes est présentée. Cette architecture a l'avantage d'être modulaire, mais elle n'est pas encore commercialisée.

4.4.3 Intérêt pour les applications physiques

Une application contextuelle programmée avec l'approche physique utilise l'espace physique comme une mémoire en le peuplant avec des données contextuelles. Le contexte physique de l'utilisateur est représenté par l'ensemble des données accessibles. Les réseaux de capteurs présentent plusieurs avantages pour transformer l'espace physique en une mémoire : ils peuvent être disséminés dans l'environnement, embarqués sur les objets physiques et sont capables de communiquer entre eux. Ils sont donc tout à fait adaptés au développement d'applications contextuelles physiques.

Une application physique utilise un contexte physique qui correspond à un ensemble de données entourant l'utilisateur et occupant la portion de l'espace physique correspondant à leur zone contextuelle. Les capteurs permettent de stocker des données dans l'espace physique. Il nous reste à définir leur zone contextuelle, pour pouvoir construire et manipuler des contextes physiques. Cette zone peut simplement correspondre à la portée de l'interface

réseau du capteur, comme pour les applications physiques présentées précédemment. Nous pouvons également envisager de créer des zones plus complexes via des protocoles de communication adaptés, ou bien en stockant une donnée sur plusieurs capteurs. Par exemple, nous pourrions stocker des références vers une donnée sur plusieurs capteurs. Modifier une référence serait équivalent à modifier la donnée. La zone contextuelle correspondante correspondrait à l'union des zones de communication des capteurs portant la donnée ou des références vers celle-ci.

4.5 Synthèse

Dans cette partie nous avons présenté différentes architectures, systèmes et applications physiques. Nous avons tout d'abord détaillé deux environnements de programmation reposant sur l'utilisation d'un espace de tuples : LIME et SPREAD. L'espace de tuples permet de résoudre simplement le problème de la sélection des données. PeerWare et les SIS sont deux modèles de communication permettant le développement d'applications contextuelles. PeerWare propose d'organiser les données de manière arborescente et de les sélectionner selon leur place dans l'arborescence. Le modèle SIS délègue aux applications la définition du mode d'organisation et de sélection des données. PERSEND stocke les données dans des bases de données et y accède avec des requêtes classiques. Avec SIDE Surfer les données sont sélectionnées d'après le profil de l'utilisateur.

L'approche physique exploite exclusivement des contextes unifiés. D'une manière générale, le contexte de l'utilisateur est représenté par l'ensemble des données accessibles. Avec SPREAD et LIME cet ensemble est constitué de tuples. Pour PeerWare les données accessibles sont représentées par une série d'arbres (cf figure 10). En ce qui concerne les SIS, le modèle représente le contexte par l'ensemble des nœuds voisins. Il peut être spécialisé ensuite par les applications. Par exemple, PERSEND le représente par l'ensemble des bases de données embarquées sur les nœuds voisins.

Actuellement les applications physiques sont déployées sur des nœuds de type PDA. Ces architectures sont adaptées pour les terminaux utilisateurs mais ne permettent pas de disséminer des informations dans l'espace physique ou d'en embarquer sur des objets facilement. De ce point de vue là les réseaux de capteurs sont prometteurs. Ils devraient permettre de résoudre ces deux problèmes dans un futur proche.

5 Comparaison des deux approches

Dans cette partie nous comparons chacune des deux approches, que nous venons de détailler, en mettant en relief leurs avantages et inconvénients respectifs.

5.1 Inconvénients de l'approche physique

L'approche physique est caractérisée par des données contextuelles disséminées dans l'environnement. La vision qu'a un calculateur du monde physique est représentée par le

contexte de l'utilisateur, elle est donc locale et réduite aux données accessibles. Une application nécessitant une vue globale n'est pas réalisable directement avec cette approche, mais nécessite l'emploi de protocoles distribués, tels que les protocoles de communication employés dans les réseaux ad-hoc. L'approche logique ne présente pas cette limitation, la plate-forme de services peut très bien contenir une vue globale du monde, stockée dans une base de données spatiales par exemple.

La seconde limitation de l'approche physique provient de la définition de la zone contextuelle des données, qui correspond au périmètre de communication du terminal associé [37]. Si nous considérons les technologies de communication actuelles, celles-ci limitent les formes réalisables à des sphères pour les ondes radio et des cônes pour les infrarouges. Les interfaces les plus courantes utilisent des ondes radio et sont de type WiFi ou BlueTooth. WiFi offre une portée de cent mètres et BlueTooth de dix ou cent mètres selon le type d'interface. Ainsi, pour une application nous sommes limités à des données ayant toutes la même forme, qui est soit une sphère de dix mètres, soit une sphère de cent mètres. Cette forme unique n'est pas adaptée à tout type de données. Pour certaines situations il serait intéressant de pouvoir définir des données plus "petites" ou ayant un forme cubique par exemple. Si nous considérons un téléviseur dans un magasin d'électroménager, sa zone contextuelle ne doit pas être trop grande afin d'être sûr que le visiteur se trouve bien devant l'appareil lorsqu'il fait partie de son contexte physique. Dans ce cas là même une sphère de dix mètres est trop grande. Avec l'approche logique ces contraintes disparaissent, il est possible d'assigner une forme quelconque aux données.

Du point de vue de la variété des applications réalisables, l'approche physique est plus limitée que l'approche logique. Par exemple, l'application d'aide au travail en collaboration, présentée précédemment (cf 3.2.1), n'est pas réalisable avec l'approche physique. Ceci vient du fait que les applications physiques exploitent un contexte physique qui est fonction de la position de l'utilisateur. Les applications qui exploitent un contexte ne dépendant pas de la position géographique sont inaccessibles à l'approche physique.

5.2 Inconvénients de l'approche logique

L'approche logique est plus polyvalente que l'approche physique. Cependant, pour les applications réalisables avec les deux approches, telles que celles nécessitant une association entre objets physiques et données, l'approche physique présente des avantages dûs à l'utilisation d'un contexte physique.

Revenons à notre exemple de guide virtuel. La première contrainte de l'approche logique est due à la nécessité d'un système de localisation. En effet la position est la seule information qui permette d'extraire les données concordant avec le contexte de l'utilisateur. L'utilisation d'un système de localisation impose un coût supplémentaire.

Ensuite, vient le problème du stockage des données, les informations sont soit stockées dans le guide, soit dans un serveur central. L'approche "tout embarqué" pose le problème des mises à jour. Comme expliqué précédemment, à chaque modification de la configuration physique de l'exposition, l'ensemble des guides doit être mis à jour pour refléter ces chan-

gements. Cette approche est viable uniquement pour des configurations statiques ou qui changent rarement. Elle est inadaptée pour des situations comprenant des objets mobiles.

De son côté, l'approche basée sur un serveur implique la mise en place d'une infrastructure de communication globale. Ensuite, le guide doit transmettre à intervalles réguliers ses coordonnées (contexte) au serveur (plate-forme de service) afin que celui-ci détermine les informations à transmettre. Cette approche n'est pas extensible facilement, lorsque le nombre de calculateurs effectuant des requêtes vers le serveur augmente, celui-ci peut devenir un goulot d'étranglement. Tout comme le dispositif de localisation, l'infrastructure de communication représente également un coût financier supplémentaire. Cette approche peut convenir pour les situations comprenant des objets mobiles, cependant les mises à jour continues des positions représentent autant de trafic pour le réseau de communication et accentuent le problème de l'extensibilité.

L'approche physique ne présente pas ces inconvénients, un système de localisation est inutile et les applications sont naturellement distribuées sur les différents calculateurs. En effet, les utilisateurs interagissent avec les calculateurs qui sont proches d'eux. Si nous supposons qu'ils sont répartis de manière homogène dans l'espace alors les requêtes seront également réparties de manière homogène sur les calculateurs. De plus, même si nous avions une plus grande concentration de personnes en un point donné, le nombre d'utilisateurs interagissant avec un calculateur est limité par la taille de sa zone contextuelle. Dans le cas d'objets mobiles, aucune mise à jour de position n'est nécessaire, les données se déplacent simplement avec les objets. Le problème de l'extensibilité ne se pose donc pas.

D'une manière générale la plate-forme de services est la source des problèmes de l'approche logique. Elle impose un coût financier supplémentaire et représente un point de congestion potentiel. Son développement et sa mise au point sont également source d'un coût supplémentaire en temps et en argent. L'absence de ce composant dans l'approche physique simplifie d'autant l'architecture logicielle des applications.

6 Conclusion

Une application contextuelle n'est pas utilisée dans les mêmes conditions qu'une application classique. Une application classique est adaptée pour les situations où la seule activité physique de l'utilisateur est de se servir d'un ordinateur. L'informatique contextuelle propose des applications adaptées aux situations où l'utilisateur est actif physiquement. Pour cela, elle cherche à diminuer les interactions avec la machine en détectant automatiquement le contexte de l'utilisateur. Une fois le contexte détecté, elle peut délivrer un service s'y conformant.

Dans cet article, nous avons présenté le problème de la programmation d'applications contextuelles et détaillé les deux grandes approches disponibles : l'approche logique et l'approche physique. L'approche logique est caractérisée par l'utilisation d'une plate-forme de services. De son côté, l'approche physique est caractérisée par l'exploitation d'un contexte physique, dispensant de plate-forme de services. Pour chacune de ces approches nous avons présenté un ensemble d'applications.

Chacune d'elles présente des avantages et des inconvénients. L'approche logique permet d'offrir une vue globale du monde aux applications et de définir une forme quelconque pour les données. Cependant, elle pose le problème de l'extensibilité, du coût financier ainsi que de la simplicité de l'architecture des applications. L'approche physique ne présente pas ces inconvénients, cependant elle doit s'affranchir de la limitation concernant la forme de données, pour pouvoir exprimer tout son potentiel. En effet, à notre connaissance il n'existe pas de technologie sans fil permettant de modeler à convenance la zone de communication. Il est donc impossible, pour l'instant, de définir de manière physique la forme des données.

Une approche hybride est envisageable afin de surmonter ce problème. En effet, les techniques employées par l'approche logique permettent d'associer une forme quelconque à une données. En les réemployant dans l'approche physique, il serait possible d'associer aux données stockées une forme autre que la sphère. Bien sûr, les données étant gérées par des calculateurs sans fil, en l'absence de réseau de communication global elles seraient incluses dans la sphère de communication. Les réseaux de capteurs semblent également offrir des possibilités permettant de définir plus finement la forme des données, mais ceci reste un problème ouvert auxquels nous consacrerons nos prochains travaux.

Références

[1] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, et M. Pinkerton. Cyberguide: A Mobile Context-aware Tour Guide. *ACM Wireless Networks*, 3:421–433, 1997.

[2] M. Banâtre et F. Weis. Systèmes d'Informations Spontanés : Problématique et Premiers Éléments de Solution. Rapport technique PI 1222, IRISA, 1998.

[3] C. Borcea, D. Iyer, P. Kang, A. Saxena, et L. Iftode. Cooperative Computing for Distributed Embedded Systems. Dans *Proc. 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, July 2002.

[4] Sergey Brin et Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[5] B. Brumitt, B. Meyers, J. Krumm, A. Kern, et S. Shafer. EasyLiving: Technologies for Intelligent Environments. Dans *Handheld and Ubiquitous Computing (HUC'00)*, September 2000.

[6] G. Chen et D. Kotz. A Survey of Context-Aware Mobile Computing Research. Rapport technique TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.

[7] K. Cheverst, N. Davies, K. Mitchell, et A. Friday. Mobile-awareness: Designing for Mobile Interactive Systems. *ACM SIGGROUP Bulletin*, 22(1):8–11, 2001.

[8] P. Couderc et M. Banâtre. Ambient Computing Applications: An Experience with the SPREAD Approach. Dans *Annual Hawaii International Conference on System Sciences (HICSS'03)*, 2003.

[9] P. Couderc et M. Banâtre. Spreading the web. Dans Springer, éditeur, *Personnal Wireless Communications (PWC'03)*, pages 375–384, 2003.

[10] P. Couderc, A.-M. Kermarrec, et M. Banâtre. Approches Adaptatives en Mobilité : une Synthèse. *Techniques et Sciences Informatiques (TSI)*, 1999.

[11] J. L. Crowley, J. Coutaz, G. Rey, et P. Reignier. Perceptual Components for Context Aware Computing. Dans *International Conference on Ubiquitous Computing (UbiComp'02)*. Springer Verlag, September 2002.

[12] G. Cugola et G. Picco. PeerWare: Core Middleware Support for Peer-To-Peer and Mobile Systems, 2001.

[13] D. López de Ipiña, P. R. S. Mendonça, et A. Hopper. TRIP: A Low-Cost Vision-Based Location System for Ubiquitous Computing. *Personal and Ubiquitous Computing*, 6(3):206–219, 2002.

[14] A. J. Demers. Research Issues in Ubiquitous Computing. Dans *Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing*, pages 2–8. ACM Press, 1994.

[15] A. K. Dey et G. D. Abowd. Towards a Better Understanding of Context and Context-Awareness. Dans *Workshop on The What, Who, Where, When, and How of Context-Awareness, Conference on Human Factors in Computing Systems (CHI'00)*, 2000.

[16] M. E. Dickover, C. L. McGowan, et D. T. Ross. Software Design Using: SADT. Dans *Proceedings of the 1977 annual conference*, pages 125–133. ACM Press, 1977.

[17] L. Doherty, L. ElGhaoui, et K. S. J. Pister. Convex Position Estimation in Wireless Sensor Networks. Dans *Proceedings of Infocom 2001*, 2001.

[18] D. Estrin, R. Govindan, J. S. Heidemann, et S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. Dans *International Conference on Mobile Computing and Networking (MobiCom'99)*, pages 263–270, 1999.

[19] L. Forlizzi, R. Hartmut Güting, E. Nardelli, et M. Schneider. A Data Model and Data Structures for Moving objects Databases. Dans *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 319–330, 2000.

[20] D. Gelernter. Generative Communication in Linda. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 7(1):80–112, 1985.

[21] L. Girod, V. Bychkobskiy, J. Elson, et D. Estrin. Locating Tiny Sensors in Time and Space: A Case Study. Dans *International Conference on Computer Design (ICCD'02)*, 2002.

[22] R. H. Güting. An Introduction to Spatial Database Systems. *The VLDB Journal - The International Journal on Very Large Data Bases*, 3(4):357–399, 1994.

[23] J. Hightower et G. Borriello. Location Systems for Ubiquitous Computing. *IEEE Computer*, 34(8):57–66, August 2001.

[24] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, et K. Pister. System Architecture Directions for Networked Sensors. Dans *Architectural Support for Programming Languages and Operating Systems (ASPLOS'00)*, pages 93–104, 2000.

[25] L.E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, et H.W. Gellersen. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. Dans *International Conference on Ubiquitous Computing (UbiComp'01)*, 2001.

[26] M. Hope, T. Chrisp, et N. Linge. Improving Co-operative Working in the Utility Industry through Mobile Context Aware Geographic Information Systems. Dans *Proceedings of the eighth ACM International Symposium on Advances in Geographic Information Systems*, pages 135–140. ACM Press, 2000.

[27] T. Schoch K. Römer. Infrastructure concepts for tag-based ubiquitous computing applications. Dans *Workshop on Concepts and Models for Ubiquitous Computing (UbiComp'02)*, September 2002.

[28] J. M. Kahn, R. H. Katz, et K. S. J. Pister. Next Century Challenges: Mobile Networking for "Smart Dust". Dans *International Conference on Mobile Computing and Networking (MobiCom'99))*, pages 271–278, 1999.

[29] Y. Kawahara, M. Minami, H. Morikawa, et T. Aoyama. Design and Implementation of a Sensor Network Node for Ubiquitous Computing Environment. Dans *Semiannual Vehicular Technology Conference (VTC2003-Fall)*, 2003.

[30] T. Kindberg. Implementing physical hyperlinks using ubiquitous identifier resolution. Dans *Proceedings of the 11th international conference on World Wide Web*, pages 191–199. ACM Press, 2002.

[31] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, et M. Spasojevic. People, Places, Things: Web Presence for the Real World, 2000.

[32] M. Langheinrich, F. Mattern, K. Römer, et H. Vogt. First Steps Towards an Event-Based Infrastructure for Smart Things. Dans *Ubiquitous Computing Workshop (PACT 2000)*, 2000.

[33] J. Li, J. Jannotti, D. De Couto, D. Karger, et R. Morris. A Scalable Location Service for Geographic Ad-Hoc Routing. Dans *International Conference on Mobile Computing and Networking (MobiCom'00)*, pages 120–130, Août 2000.

[34] I. MacColl, D. Millard, C. Randell, et A. Steed. Shared Visiting in EQUATOR City. Dans *International Conference on Collaborative virtual environments*, pages 88–94. ACM Press, 2002.

[35] M. P. G. Oliveira, E. Bauzer M., et C. A. Davis. Planning the Acoustic Urban Environment: a GIS-centered approach. Dans *Proceedings of the seventh ACM International Symposium on Advances in Geographic Information Systems*, pages 128–133. ACM Press, 1999.

[36] J. Pascoe. Adding Generic Contextual Capabilities to Wearable Computers. Dans *International Symposium on Wearable Computers (ISWC'98)*, pages 92–99, 1998.

[37] J. Pauty, M. Banâtre, et P. Couderc. Logical versus Physical Programming for Ubiquitous Computing. Dans *Workshop on Intelligent Solutions in Embedded Systems (WISES'03)*, June 2003.

[38] E. Peytchev et C. Claramunt. Experiences in Building Decision Support Systems for Traffic and Transportation GIS. Dans *Proceedings of the ninth ACM International Symposium on Advances in Geographic Information Systems*, pages 154–159. ACM Press, 2001.

[39] G. P. Picco, A. L. Murphy, et G.-C. Roman. LIME : Linda Meets Mobility. Dans *International Conference on Software Engineering (ICSE'99)*, pages 368–377, 1999.

[40] Gian Pietro Picco, Amy L. Murphy, et Gruia-Catalin Roman. Developing mobile computing applications with LIME. Dans *International Conference on Software Engineering (ICSE'00)*, pages 766–769, 2000.

[41] J. M. Rabaey, M. J. Ammer, J. L. da Silva Jr, et D. P. S. Roundy. PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking. *IEEE Computer*, pages 42–48, 2000.

[42] J. Rekimoto et Y. Ayatsuka. Cybercode: Designing augmented reality environments with visual tags. Dans *Proceedings of Designing Augmented Reality Environments (DARE'00)*, pages 1–10. ACM Press, 2000.

[43] J. Rekimoto et K. Nagao. The World Through the Computer: Computer Augmented Interaction with Real World Environments. Dans *ACM Symposium on User Interface Software and Technology (UIST'95)*, pages 29–36, 1995.

[44] G. Roussos, J. Tuominen, L. Koukara, O. Seppala, P.s Kourouthanasis, G. Giaglis, et J. Frissaer. A Case Study in Pervasive Retail. Dans *Proceedings of the second international workshop on Mobile commerce*, pages 90–94. ACM Press, 2002.

[45] A. Schmidt, M. Beigl, et H.-W. Gellersen. There is more to Context than Location. *Computers and Graphics*, 23(6):893–901, 1999.

[46] S. Tilak, N. Abu-Ghazaleh, et W. Heinzelman. A Taxonomy of Wireless Microsensor Network Models. *ACM Mobile Computing and Communications Review (MC2R 2002)*, 6(2):28–36, 2002.

[47] D. Touzet, F. Weis, et M. Banâtre. Accès à l'information en ubiquité numérique. Rapport technique P.I. 1460, IRISA, 2002.

[48] A. Troel, M. Banâtre, P. Couderc, et F. Weis.. Predictive scheme for proximate interactions. Dans *International Conference on Distributed Computing Systems Workshops (ICDCSW'01)*, 2001.

[49] R. Want, K. P. Fishkin, A. Gujar, et B. L. Harrison. Bridging Physical and Virtual Worlds with Electronic Tags. Dans *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 370–377. ACM Press, 1999.

[50] M. Weiser. Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM, Back to the Real World, Special issue on Computer Augmented Environments*, 36(7):75–84, 1993.

[51] O. Wolfson, B. Xu, S. Chamberlain, et L. Jiang. Moving Objects Databases: Issues and Solutions. Dans *10th International Conference on Scientific and Statistical Database Management*, pages 111–122, 1998.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399