Département Informatique
Année 2004/2005

Rapport de Synthèse

Environnements des services pervasifs gérés par l'intention de l'utilisation

Pascal BIHLER

LIRIS UMR 5205
INSA, Campus de la Doua
Bâtiment Blaise Pascal (501)
20, avenue Albert Einstein
69621 VILLEURBANNE CEDEX

Lionel Brunie lionel.brunie@insa-lyon.fr
Vasile-Marian Scuturici marian.scuturici@insa-lyon.fr

Résumé
L'introduction des systèmes pervasifs dans la vie quotidienne n'est pas seulement un grand pas pour les utilisateurs, mais aussi pour les développeurs de logiciels. Les systèmes pervasifs proposent de remplacer les interfaces d'interaction traditionnelles comme le clavier ou la souris par d'autres modes d'interaction plus intuitifs. Le défi pour l'intergiciel d'un système pervasif est alors de capturer et modéliser l'intention de l'utilisateur et d'en résoudre les ambiguïtés. Dans le cadre de ce PFE, un Langage de Requêtes d'Actions de Services Pervasifs (PsaQL) est développé. PsaQL formalise la description de l'intention de l'utilisateur en utilisant des services pervasifs composés. Un autre défi pour ce PFE était la définition d'une manière de traduire une intention de l'utilisateur en une action exécutable ainsi que la proposition des algorithmes qui réalisent cette traduction. Dans le cadre de PERSE, un Environnement des Services Pervasifs développé par l'équipe de LIRIS et des considérations pour implémenter ce processus sont données, ainsi que les mesure d'évaluation de tels algorithmes. Un prototype de PERSE qui inclut un module d'interprétation de l'intention de l'utilisateur est développé et des résultats de tests de performances avec ce prototype sont prises.

Mots clefs
Systèmes pervasifs, calcul ubiquitaire, services, intention de l'uilisateur, PERSE, PsaQL

Abstract
The introduction of pervasive computing environments in everyday life will not just be a big step for users, but also for application designers. The well defined interaction interfaces "keyboard" and "mouse" will make place for other, more intuitive ways of interaction. It is the challenge for a pervasive system middleware to capture and model the user's intention in a smart way and to solve ambiguousness in the user's expression of a pervasive action. In the scope of this PFE the Pervasive Service Action Query Language (PsaQL) has been developed. PsaQL formalize the description of a user intention using composed pervasive services. Another challange fort his PFE was the definition of a way of translating the user intention into an executable action and propose algorithms performing this translation. Considerations to implement this process have been made within the scope of PERSE, a pervasive service environment developed by a LIRIS research group and general evaluation metrics for such algorithms have been defined. A prototype of PERSE including the user intention interpretation module was developed and benchmark results have been taken fort his prototype interpreter.

Key words
Pervasive Computing, Ubiquitious Computing, Services, User Intention, PERSE, PsaQL

Introduction

Dans le laboratoire LIRIS de Lyon, une équipe de Lionel Brunie s'occupe de la réalisation d'un monde pervasif : dans l'espace vital et professionnel sont des éléments (dispositifs) embarqués qui ont de la capacité de calculer et communiquer. Ces « ordinateurs » ne sont plus reconnaissables comme machines, mais plutôt comme des parties habituelles d'un environnement intelligent.[1]
Si l'utilisateur ne reconnaissait plus les ordinateurs comme machines techniques, aussi la mode d'interagir avec ces ustensiles doit changer. Dans la vision de l'équipe de Lionel Brunie et Vasile-Marian Scuturici, où j'ai effectué mon PFE, l'interaction d'un utilisateur avec un équipement pervasif est intuitive et non-intrusive : L'utilisateur exprime son intention, par exemple avec la voix ou avec des gestes, et l'environnement intelligent réagit avec une action raisonnable.
En plus, une deuxième caractéristique d'un environnement pervasif doit être de ne plus seulement réagir à des ordres de l'utilisateur, mais plutôt agir « proactif » (cf. figure 1). Dans la compréhension de l'équipe, un système agit de façon proactive, s'il comprend l'intention de l'utilisateur en observant son contexte, s'il prend en compte des actions historiques pour choisir une action appropriée et s'il propose et exécute des actions raisonnables pour l'utilisateur au bon moment.
Dans le laboratoire LIRIS, nous avons développé un environnement des services pervasifs, PERSE. Dans un environnement de services pervasifs, les appareils du système pervasif (par ex. un ANP, un senseur, un projecteur) propose des services (applications qui effectuent un travail très particulier). L'intergiciel de l'environnement combine ces services et organise le flux de données entre eux pour réaliser une action complexe.
Dans le cadre de mon PFE c'était le défi d'intégrer l'intention de l'utilisateur dans le système PerSE. Pour cela, il était important de définir formellement les différentes étapes de venir d'une intention exprimée diffusément par l'utilisateur vers une action exécutable. Pour cela, il faut utiliser des informations contextuelles et l'historique d'exécution du système. Pour décrire précisément l'intention de l'utilisateur, j'ai élaboré un langage formel, PsaQL. Ensuite, les algorithmes de traduction l'intention de l'utilisateur dans une action exécutable étaient réalisés dans une forme prototype et inclus dans l'intergiciel du PERSE. Cela était accompagné par la définition d'un modèle de classe pour l'implantation des actions et par la définition d'un monnayage de XML pour échanger les données. Finalement, j'ai élaboré une possibilité de mesure de la performance d'une implantation en proposant une grille des mesures, important pour assurer la qualité pendant le développement d'un système.

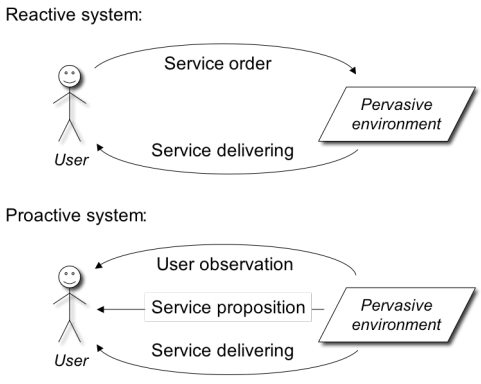

Figure 1 : La comparaison des systèmes réactifs et proactifs. Des systèmes réactifs attendent un ordre explicit de l'utilisateur pour lancer une action où les systèmes proactifs essaient de dériver une action au base des informations contextuelles.

[1] La montre, par exemple, est aujourd'hui aussi une partie de la vie quotidienne et n'est plus sentie comme intrusive.

Travaux connexes sélectionnés

La plateforme du El Kathib et al. [KBS04] est un travail particulièrement intéressant, car les auteurs essaient aussi d'utiliser la satisfaction de l'utilisateur comme critère d'évaluation. Ils adaptent des fluxs multimédias dynamiquement avec un graphe comme modélisation interne du système. En différence de la solution élaborée pendant ce PFE, El Kathib travail avec des structures du réseaux statique, comme dans l'Internet, et pas avec des environnements dynamique comme donnés dans les systèmes pervasifs. La flexibilité de son approche permis peut-être d'utiliser la plateforme aussi avec les réseaux dynamiques.
Un autre travail, qui est situé proche de notre problématique, est le système de M. Vallée [VRV05]. Dans son plateforme il propose comme PERSE une combinaison de services pervasifs pour répondre aux besoins de l'utilisateur avec des actions. En commencent avec un plan abstrait, l'algorithme de combinaison en utilisant des informations contextuelles choisit un plan détaillé d'une base des données, qui convient le mieux à la situation.
Le travail de ce PFE était très fortement inspiré par l'idée des Semantic Web Services (cf. par ex. [MSZ01]). Des références supplémentaires pour ce travail sont données dans [Bih05].

L'environnement PERSE

Dans un environnement des services pervasifs collaborent des applications indépendantes, situées dans des dispositifs pervasifs divers pour effectuer une action complexe. C'est le défi de l'intergiciel du système de gérer cette collaboration d'une façon raisonnable.
Un dispositif appartient à un environnement PERSE s'il contient un service spécial que nous appelons *PerseBase* (cf. figure 2). Ce méta service permet de gérer d'autres services existants sur le dispositif. Une base peut communiquer avec les bases voisines pour acquérir ou offrir des informations utiles pour la gestion des actions de l'environnement. Ces connexions sont réalisées d'une manière ad-hoc. L'environnement ne comporte pas de composants ayant un rôle central, et chaque base peut fonctionner de manière indépendante.
Dans ce PFE, une manière de traduire une expression d'intention de l'utilisateur[2] en action exécutable a été développé[3] : L'intention de l'utilisateur est donnée dans une *action partielle* (par ex. sous la forme de *PsaQL*), une description qui contient plus ou moins précis défini le début et le fin des données et peut-être des étapes de traitement en milieu. Après, toutes les actions possibles sont calculées en formant le *graphe d'actions* et finalement la meilleure action est sélectionnée en utilisant les informations contextuelles et l'histoire de l'exécution (cf. figure 3).
Pour figurer le processus de transformation d'une intention de l'utilisateur en action, cet exemple peut aider (cf. figure 4) :
Une personne entre dans une salle pour montrer une présentation. En utilisant le système PERSE, son ordinateur portable ou son APN entre en contact avec le serveur

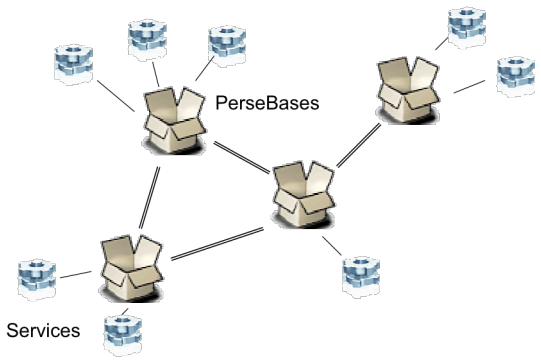

Figure 2 : L'architecture de PerSE : Plusieurs PerseBases gèrent des services et sont connectés avec des autres PerseBases.

[2] La capture de cette intention n'été pas traité dans le cadre de ce PFE.
[3] Une définition formelle pour les termes et les idées introduit dans ce qui suit est donné dans [Bih05], 5.1.

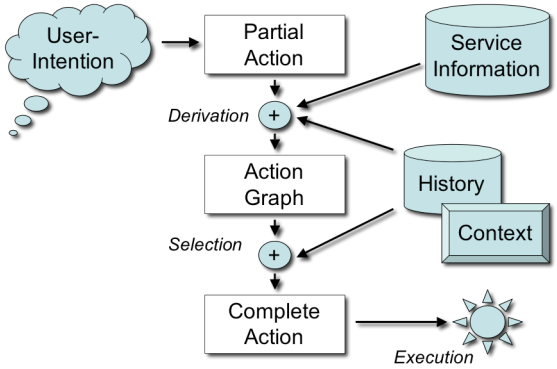

Figure 3 : Une intention de l'utilisateur est transformée dans une action complète par sélection de la meilleure action de l'ensemble de toutes les actions possibles (graphe d'action) en utilisant des informations contextuelles et l'histoire de l'exécution.

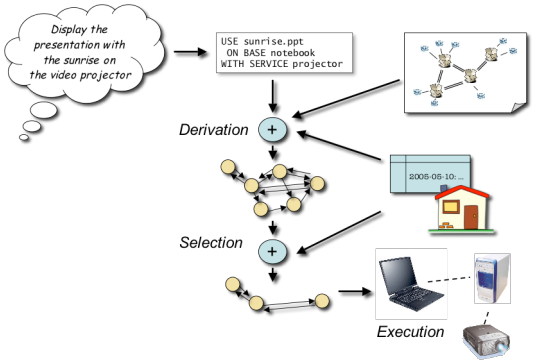

Figure 4 : Une présentation est montrée sur un projecteur

embarqué invisible dans la salle, qui est connecté à un projecteur. Suite à une commande tapée ou dite par la personne (« Montre la présentation avec le couché de soleil sur le projecteur »), le système établit une connexion entre le dispositif mobile et le projecteur et montre la présentation sans charger l'utilisateur avec des détails techniques.

La modélisation de l'intention de l'utilisateur

Si l'utilisateur exprime son intention, il donne cela normalement dans une forme, qui ne peut pas directement être traitée par le système de gestion de l'environnement pervasif. Mais, même si cette expression est traduite dans une forme textuelle, il manque encore la formalisation.
Pour exprimer une action partielle, le langage *PsaQL (Pervasive Service Action Query Language)* était développé dans le cadre de ce PFE. Il signifie pour un environnement des services pervasifs le même que SQL [Dat86] pour le système de bases de données relationnelles. C'est la raison pour laquelle les expressions aussi se ressemblent :

```
USE sunrise.ppt
 ON BASE notebook
 WITH SERVICE projector
```

La définition en BNF est comme suivante :

```
<partial_action> ::=
    USE <action_part> [<ext_action>]
<ext_action> ::=
    WITH <action_part> [<ext_action>]

<action_part> ::= <attr_constr_def>
                  [ FOR <service_constr_def>]
                  [ ON <base_constr_def>] |
                <service_constr_def>
                  [ ON <base_constr_def>] |
                <base_constr_def>

<base_constr_def> ::=
    BASE <base_constraint>[ AS <name>]
<base_constraint> ::=
    <name> | LIKE "<part_name>"

<service_constr_def> ::=
   SERVICE <service_constraint>[ AS <name>]
<service_constraint> ::=
   <name> | LIKE "<part_name>"

<attr_constr_def> ::= (<name> |
    LIKE "<part_name>")[ AS <name>]

<name> ::= {<'a'-'z','A'-'Z','0'-'9','_'>}
<part_name> ::= {<'a'-'z','A'-'Z','0'-'9',
             '^','$', '(',')','[',']','.',
             '+','*','?', ...>}
```

Actuellement, la déclaration LIKE est utilisée pour définir une expression régulière. Un prototype pour montrer le processus de traduire une expression PsaQL dans une action complète était développé, accessible à http://liris.wh4f.de/perse.

La traduction d'une action partielle dans une action complète

La traduction d'une action partielle dans une action complète se compose de trois étapes :

1. Traduire la requête (donnée par ex. en PsaQL) dans une représentation interne d'une action partielle.
2. Élargir l'action partielle à un graphe d'action en utilisant des descriptions des services connaissent dans les réseaux, l'histoire d'exécution, des informations contextuelles et des stratégies heuristiques.
3. Sélectionner la meilleure action dans le contexte actuel comme action complète.

Tandis que la première étape est seulement une analyse syntaxique, les deux autres étapes doit être expliqués plus détaillés.

Pour décrire formellement le traitement des données dans le système de traduction, un formalisme était défini dans [Bih05], chapitre 5.1. Les termes définis là sont entre autres (ici simplifiés) :

- *Action partielle :* Une liste de paires *(Base, Service)*, ou un des deux peut être indéfini.
- *Graphe de services :* La paire d'un ensemble de paires *(Base, Service)* et un ensemble des combinaisons de ces paires représentant les connexions entre eux.
- *Solution :* Un graphe de services connexe qui contient un service pour chaque partie de l'action partielle.
- *Graphe d'action :* L'union de toutes les solutions possibles.
- *Action complète :* Le sous graphe du graphe d'action, qui est une solution est qui a les coûts minimaux de tous le solutions dans le contexte actuel.

Dans un environnement des services pervasifs d'une taille bordée, le calcul de l'action complète peut être effectué dans une manière exhaustive. L'algorithme développer pour ce cas est documenté dans [Bih05], page 14. L'idée de ce algorithme est la suivante : à partir du graphe d'action, les connexions sont supprimés étape par étape. Le sous graphe avec les coûts minimaux, qui est encore une solution valide, est pris comme action complète.

Dans les cas, où l'environnement de services pervasifs est plus complexe, cet algorithme n'est plus applicable, car il montre une complexité exponentielle. Pour avoir une complexité linéaire, nous avons introduit une heuristique.

Cet algorithme (pour un codage exemplaire, cf. aussi [Bih05], page 14) est basé sur une colorisation de nœuds dans le graphe d*'action :

- *Nœuds noirs :* Les nœuds qui sont uniquement définis par une partie de l'action partielle.
- *Nœuds rouges :* Les nœuds où l'on trouve une partie de l'action partielle correspondent non injective.
- *Nœuds blancs :* Les nœuds, qui ne correspondent pas à une partie de l'action partielle.

Chaque nœud comporte seulement une couleur avec l'ordre *noir* > *rouge* > *blanc*.

L'action complète, qui doit être calculée par l'algorithme, contient tous les nœuds noirs, quelques nœuds rouges et peut-être des nœuds blancs. L'heuristique pour trouver l'action complète procède comme suivant (cf. figure 5) :

Dans une première étape, les distances entre les nœuds rouges et le nœud noir le plus proche sont calculés. À partir du nœud rouge avec la distance minimale, tous les nœuds rouges sont examinés dans l'ordre de leur

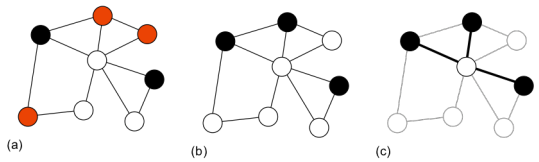

Figure 5 : À partir d'une action partielle avec trois parties (définissent deux nœuds noirs et trois nœuds rouges) une action complète est calculée.

distance : Si il y une partie de l'action partielle correspondante, qui n'est pas déjà couverte par un nœuds noir, le nœuds actuel est coloré en noir aussi. Les nœuds restants sont colorés en blanc. Enfin, l'arbre enjambant qui contient tous les nœuds noirs avec les coûts minimaux est pris comme action complète.

L'implémentation des graphes d'action

Pour l'implémentation des graphes d'action et des actions complètes, une structure de données interne était à développer. Chaque graphe de services est connexe, donc c'est pratique de modéliser les graphes et les actions complètes (qui sont aussi des graphes de service) comme un ensemble de canaux, qui connecte les points de connexion des services.
La modélisation est orientée objet pour bénéficier de la réutilisation des objets et de l'identité unique de chaque objet. La structure des donnés est ébauchée dans la figure 6. Les classes utilisées sont :

- *Address :* Une combinaison d'un type et une chaîne de caractère de l'adresse, par ex. `"ipv4"/"170.20.0.9:80"`.
- *Base :* Le représentant d'un PerseBase, contient un ou plusieurs *Adresses* et gère des *Services.*
- *Service :* Un service propose des *Ports* pour l'entrée et la sortie des données.
- *Port :* Représentation pour une entrée ou une sortie des données d'un *Service*, contient des informations sur le type de données échangées et la direction (« in », « out », « inout »)
- *Element :* La combinaison d'un *Service* avec des informations d'adaptation (*Attributes*).
- *Attribute :* Une chaîne de caractère pour adapter un *Service* aux besoins de l'action spécifique.
- *Stub :* La combinaison d'un *Element* avec un *Port.*
- *Channel :* Le flux de données entre deux *Stubs.*
- *Action :* Un ensemble des *Channels*.

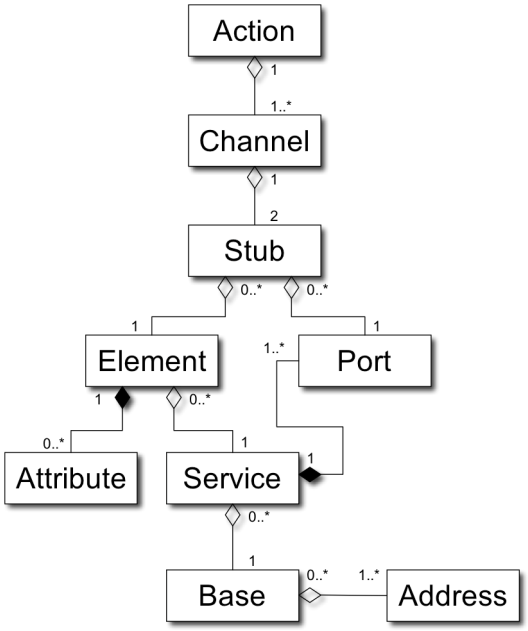

Figure 6 : Le diagramme des classes pour la représentation interne des graphes de service.

Les algorithmes, implantés dans les PerseBases, qui calculent l'action complète utilisent une base des données locales avec les descriptions des services. Un PerseBase connaît à priori seulement ces propres services. Pour accéder aux descriptions distantes, une interface http a été définit avec les types des requêtes suivantes :

- *AllObjects* : Rendre une liste de tous les PerseBases voisins.
- *services* : Rendre une liste de tous les services locaux avec quelques métas informations.
- *serviceDescription* : Rendre une description détaillée d'un service avec les informations sur les ports existants.

La communication est basée sur des fichiers XML qui sont échangés (cf. [Bih05], A^{xe} A). L'utilisation des ontologies comme OWL [MvH04] est planifiée pour déterminer l'interopérabilité des services.
Pour échanger l'action complète entre des modules divers d'un PerseBase ou entre des entités diverses, un format XML a été développé qui contient toutes les informations important pour l'exécution d'une action complète. Un exemple se trouve dans [Bih05], page 18.

Mesurer les algorithmes

Pour comparer des algorithmes et leurs implémentations ainsi qu'assurer la qualité pendant le développement du PERSE et son utilisation, les critères suivants étaient identifiés :

- *La satisfaction de l'utilisateur* est le critère le plus important, mais aussi le plus difficile à chiffrer.
- *Le temps d'exécution* détermine directement la satisfaction de l'utilisateur.
- *La transmission des données aux réseaux* avec les sous critères *vitesse de réseau, taille des informations transférées* et *distance du transfert.*
- *La scalabilité*, si *le nombre des services* ou *la longueur de la requête* augmente.
- *Des contraintes des systèmes pervasifs* comme l'utilisation économique du processeur ou du mémoire.

La scalabilité de notre prototype a été estimée dans des expérimentations, les résultats sont décrits en [Bih05].

Implémentation du proof-of-concept

Pour démontrer l'applicabilité de notre solution, le pilotage d'une présentation avec un ANP en utilisant le système PERSE a été choisit. Pendant les services client et serveur étaient développés par deux stagiaires, le module de pris en compte l'intention de l'utilisateur a été inclut dans le prototype de PERSE. Dans le résultat, c'est possible de lancer une présentation avec une requête en PsaQL qui connecte un ANP avec le serveur du projecteur, un utilisant une ou plusieurs passerelles. Les passerelles adaptent l'image du projecteur pour l'ANP d'où les commandes pour piloter la présentation sont transmises vers le serveur (cf. figure 7).

Gestion du projet

Le projet se composait de plusieurs sous projets, qui ont effectués en parallèle et doivent être synchronisé : le *Projet de Fin d'Etudes*, la *synthèse bibliographique*, le *stage et mémoire de master* et l'applications pour des conférences internationales. La durée du projet était de février jusqu'à septembre, jusqu'à six personnes étaient impliquées (le chef de projet, le tuteur, le chef d'équipe, un encadreur et deux stagiaires). Le nombre de taches à coordonner était environ 40, trois bornes étaient définies pour le projet.
La gestion du projet a eu du succès, parce que tous les participants étaient capables de travailler individuellement ou en sous-groupes. La coordination globale avec le système du courriel et des réunions a aussi bien fonctionné. Seulement petits délais ont apparu, ils étaient provoqués par des vacances ou la surcharge d'un participant à la cause des autres projets indépendants. Quand même, les

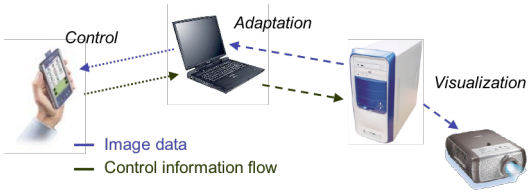

Figure 7 : Un proof-of-concept de PERSE - une présentation est contrôlée avec un APN.

délais n'ont pas conduit dans un dépassement de aucune échéance.

Conclusions

Dans le cadre de ce projet une possibilité de prendre en compte l'intention de l'utilisateur dans la plate-forme PERSE a été développée. Les fondements théoriques ont été élaborés ainsi que l'implémentation pratique a été réalisée dans la forme d'un prototype.
Le Projet de Fin d'Études situé dans un groupe de recherche a été très intéressant pour moi, car il m'a permis d'appréhender la recherche, mais aussi les problèmes d'une mise en pratique et les questions éthiques, qui se posent dans le contexte de l'informatique pervasive. L'actualité de sujet était un facteur majeur et le placement des résultats sur des conférences internationales est une confirmation d'avoir pris une bonne décision.

Glossaire :

ANP : Assistant Numérique Personnel (angl. PDA)
BNF : Backus-Naur Form, une syntaxe méta pour décrire des grammaires
LIRIS : Laboratoire d'InfoRmatique en Images et Systèmes d'information
OWL : Web Ontology Language
PERSE : Pervasive Service Environment
XML : Extended Markup Language

Références bibliographiques :

[Bih05] Pascal Bihler
How to take the user on his word. Expressing and interpreting user intention in a pervasive service environment. Mémoire de master, INSA de Lyon 2005

[Dat86] C. J. Date
A guide to the SQL standard, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[KBS04] Khalil El-Khatib, G. v. Bochmann, A. El Saddik
A distributed content adaptation framework for content distribution networks (en ligne: http://beethoven.site.uottawa.ca/dsrg/PublicDocuments/Publications/ElKh04c.pdf), 2004, dernière visite: 15/06/2005.

[MSZ01] Sheila A. McIlraith, T. Cao Son, H. Zeng
Semantic web services, IEEE Intelligent Systems, vol. 16, IEEE Educational Activities Department, Mai 01, p. 56–53

[MvH04] Deborah L. McGuinness, Frank van Harmelen,
OWL Web Ontology Language Overview (en ligne: http://www.w3.org/TR/2004/REC-owl-features-20040210/), 10/02/2004, dernière visite: 15/06/2005.

[VRV05] Mathieu Vallée, F. Ramparany, L. Vercouter
Composition flexible de services d'objets communicants, UBIMOB 05, 31 mai - 3 juin 2005.