Analyse des mécanismes de découverte de services avec prise en charge du contexte et de l'intention

Salma Najar, Manuele Kirsch Pinheiro, Luiz Angelo Steffenel, Carine Souveyet

▶ **To cite this version:**

Salma Najar, Manuele Kirsch Pinheiro, Luiz Angelo Steffenel, Carine Souveyet. Analyse des mécanismes de découverte de services avec prise en charge du contexte et de l'intention. 8èmes Journées Francophones Mobilité et Ubiquité (Ubimob 2012), Jun 2012, Anglet, France. pp.210-221. hal-00740068

HAL Id: hal-00740068
https://paris1.hal.science/hal-00740068

Submitted on 9 Oct 2012

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.

Analyse des Mécanismes de Découverte de Services avec Prise en Charge du Contexte et de l'Intention

Salma Najar, Université Paris 1 – Panthéon Sorbonne
Manuele Kirsch Pinheiro, Université Paris 1 – Panthéon Sorbonne
Luiz Angelo Steffenel, Université de Reims Champagne-Ardenne
Carine Souveyet, Université Paris 1 – Panthéon Sorbonne

Résumé : *Avec la démocratisation des dispositifs et des réseaux mobiles, les systèmes d'information deviennent pervasifs. Les utilisateurs de ces systèmes doivent désormais évoluer dans de véritables espaces de services, dans lesquels plusieurs services sont offerts. Afin d'améliorer la transparence des systèmes d'information pervasifs, nous proposons une nouvelle approche pour ces systèmes, à la fois sensible au contexte et intentionnelle. Dans cette approche, les services offerts par ces systèmes sont proposés afin de satisfaire à une intention, laquelle correspond à l'expression d'un but utilisateur, dans un contexte donné. Pour valider cette approche, nous développons actuellement une plate-forme de découverte de services basée sur une extension de OWL-S qui intègre ces notions. Grâce à des résultats expérimentaux, nous analysons l'impact de l'usage de ces notions de contexte et d'intention dans la sélection de services et démontrons l'intérêt de notre approche dans la découverte des services.*

Mots-clés : *découverte de services, intention, sensibilité au contexte, systèmes d'information pervasifs*

1 Introduction

Aujourd'hui, le développement des technologies mobiles a changé l'accès aux Systèmes d'Information (SI). Au lieu d'être au premier plan, les Systèmes d'Information résident de plus en plus en arrière-plan, surveillant les activités des utilisateurs, assemblant les informations et intervenant lorsque cela est nécessaire [10]. Ceci marque l'émergence des Systèmes d'Information Pervasifs (SIP), lesquels visent notamment à améliorer la productivité des utilisateurs grâce à la disponibilité des SI partout et à n'importe quel moment.

Les SIP représentent une nouvelle classe de SI. Contrairement aux SI traditionnels, les SIP doivent gérer une multitude de dispositifs hétérogènes et veiller à la bonne interaction entre l'utilisateur et l'environnement physique [10]. L'évolution des SI en SIP dépasse la simple évolution des services, car les utilisateurs, dans les SIP, doivent naviguer dans un véritable espace de services à la recherche de ceux qui s'adaptent au mieux à leurs besoins. Cette notion d'espace de services ne doit pas se limiter aux services logiques traditionnellement offerts par les SI, mais elle doit intégrer également les services dits "physiques", i.e., les services embarqués dans l'environnement physique.

Les systèmes pervasifs doivent se caractériser par leur transparence, selon Weiser [23]. Vingt ans plus tard, force est de constater que nous n'avons pas atteint l'environnement homogène et invisible décrit par Weiser [23]. Ceci ne veut pas dire pour autant que l'informatique ubiquitaire ne soit pas déjà une réalité. Selon Bell et Dourish [2], l'informatique ubiquitaire a tout simplement pris une autre forme que celle attendue par Weiser, en intégrant les dispositifs mobiles comme élément central du quotidien. En réalité, nous interagissons avec une panoplie de dispositifs et de services, offerts par l'ensemble des SI qui nous entoure. Alors que des grands efforts ont été concentrés sur l'adaptation au contexte, surtout à la localisation et aux dispositifs utilisés, nous constatons aujourd'hui les limitations de ces approches, notamment une certaine surcharge des utilisateurs due aux "faux positifs". Les utilisateurs se voient proposer plusieurs implémentations pour un même service, sans avoir pour autant le bagage nécessaire pour comprendre ces implémentations, ce qui nuit la transparence d'utilisation de ces systèmes. La clé du succès serait donc d'offrir à l'utilisateur le service qui satisfait ses besoins, sans qu'il soit forcé d'apprendre ou de connaître les détails concernant l'implémentation ou les contraintes des dispositifs utilisés.

Nous pensons que seulement une approche centrée sur l'utilisateur sera capable d'apporter des services adaptés au contexte d'utilisation, tout en gardant un niveau de transparence convenable. C'est dans ce soucis que nous proposons un mécanisme de découverte de services guidé non seulement par le contexte d'utilisation, mais également par son intention, apportant ainsi une vision

centrée sur l'utilisateur aux SIP. La notion d'intention, qui porte plusieurs définitions, peut être traduite comme le but qu'un utilisateur souhaite atteindre sans avoir à spécifier comment y parvenir [9] ou encore comme le but à atteindre pour mener à bien un processus, celui-ci étant composé d'une séquence de sous-buts et de stratégies pour l'atteindre [5].

Dans ce travail, nous considérons que l'intention représente les exigences formulées par l'utilisateur, qui sait ce qu'il attend d'un service mais qui ne sait pas indiquer comment y parvenir. Ces intentions émergent dans un contexte d'utilisation précis. La notion de contexte peut être définie comme toute information capable de caractériser une entité (une personne, un endroit, un objet) considérée comme pertinente pour l'interaction entre l'utilisateur et l'application [6]. Il s'agit d'un élément important dans le processus d'adaptation d'un système à l'utilisateur, processus auquel nous souhaitons ajouter la notion d'intention. Nous pensons que la satisfaction des intentions de l'utilisateur dans un SIP dépend du contexte dans lequel se trouve l'utilisateur, mais aussi que le contexte impacte la manière dont les intentions sont satisfaites, tout comme le contexte impacte le choix des services qui seront exécutés. Cette relation peut être exploitée dans la découverte des services : grâce à l'observation de l'intention et du contexte, un mécanisme plus précis peut être mis au point, offrant aux utilisateurs les services les plus adaptés à leurs besoins.

Dans cet article nous présentons un mécanisme de découverte de services à la fois contextuelle et intentionnelle. Nous présentons une implémentation Java pour ce mécanisme et analysons les résultats des expérimentations menées notamment par rapport à la scalabilité, à la précision et au rappel. Ces trois critères nous semblent particulièrement importants, puisqu'ils démontrent non seulement la faisabilité de l'approche, mais aussi son intérêt pour la découverte de services plus appropriés aux utilisateurs.

Cet article s'organise comme suit : la section 2 introduit brièvement l'état de l'art concernant la découverte de services guidée par le contexte ou par l'intention. La section 3 détaille le mécanisme proposé et son implémentation, alors que la section 4 analyse les résultats de son expérimentation. La section 5 propose nos conclusions et travaux futurs.

2 État de l'Art

La dernière décennie a témoigné d'un important changement de philosophie dans les Systèmes d'Information (SI). Avec les nouvelles technologies, les SI deviennent de plus en plus complexes et sont utilisés dans des situations très variées. Les Systèmes d'Information Pervasifs (SIP) constituent ainsi une classe émergente de SI dans lesquels les technologies de l'information sont graduellement intégrées à l'environnement physique, de manière à répondre aux besoins et aux envies de l'utilisateur à n'importe quel moment [10]. Parmi les principales caractéristiques de cette nouvelle classe de SI, Kouruthanassis [10] observe que, au delà de l'hétérogénéité des dispositifs, les SIP doivent également offrir de la sensibilité au contexte, issue de la possibilité de collecter, d'analyser et de gérer les données environnementales grâce aux dispositifs utilisés.

La sensibilité au contexte est la base pour différentes approches de découverte de service [3][21] [22][24]. Parmi ces approches, Toninelli et al. [21] considèrent que, dans les environnements pervasifs, les utilisateurs ont besoin de services sensibles au contexte, adaptés à des paramètres tels que la localisation, l'environnement d'exécution, etc. Ces auteurs proposent un mécanisme de découverte de services personnalisé basé sur la sémantique, dont l'objectif est d'intégrer une représentation sémantique des données contextuelles et l'analyse des correspondances au mécanisme de sélection.

De manière similaire, Ben Mokhtar et al. [3] proposent un mécanisme de mise en correspondance sémantique des services, guidé par le contexte. Leur proposition inclut un langage pour la spécification sémantique de services fonctionnels et non-fonctionnels appelé EASY-L. Xiao et al. [24] considèrent également les services sensibles au contexte d'un point de vue sémantique. Leurs travaux utilisent les ontologies afin de développer l'importance des paramètres de contexte d'utilisation et ainsi d'identifier automatiquement les relations entre les différentes valeurs de

contexte. Grâce à ces relations, leur mécanisme est capable d'identifier et de sélectionner les services potentiellement nécessaires à l'utilisateur. Contrairement aux auteurs précédents, Vanrompay et al. [22] considèrent la description de contexte comme un graphe, proposant des mesures de similarité pour comparer le contexte courant de l'utilisateur à celui associé au service.

Ces auteurs [3][21][22][24] illustrent une thématique de recherche majeure, laquelle met en avant l'importance du contexte lors de la découverte des services. Cependant, ces mécanismes requièrent de la part de l'utilisateur des connaissances techniques complexes afin de décrire leur environnement et de choisir un service parmi plusieurs choix suggérés, alors que normalement ces utilisateurs demandent simplement un service qui répond à leurs besoins.

Une approche différente est donc proposée par des travaux tels que [1][9][14][19]. Cette approche, dite guidée par l'intention, prône l'importance des besoins de l'utilisateur lors du choix des services. Parmi ces travaux, [9][19] proposent une architecture orientée services basée sur une perspective intentionnelle. Cette architecture propose la notion de service intentionnel, laquelle représente un service orienté par le but qu'il doit satisfaire au lieu des fonctionnalités qu'il doit exécuter. Cette notion de service intentionnel représente une alternative pour combler l'écart entre les descriptions techniques de bas niveau, liées à l'implémentation des services, et les descriptions de haut niveau qui expriment les besoins métier des utilisateurs de ces services. A partir du modèle de services intentionnel proposé par [19], Aljoumaa et al. [1] présentent une solution de découverte basée sur les ontologies qui permet la mise en correspondance entre les besoins utilisateur formulés avec des expressions métier et les intentions des services publiées dans un annuaire étendu.

Toujours dans une approche intentionnelle, Mirbel et al. [11] proposent l'utilisation de modèles et de langages sémantiques afin d'enrichir la description des besoins et ainsi de proposer des méthodes d'analyse pour la découverte de services basés sur l'intention. Les requêtes des utilisateurs sont exprimées avec des technologies du Web Sémantique, ce qui permet d'aider les utilisateurs appartenant à une communauté d'experts de trouver les services qui correspondent à leurs besoins. Olson et al. [14] défendent également l'utilisation de l'intention pour décrire les services selon un nombre arbitraire de niveaux d'abstraction. Selon ces auteurs, un algorithme de raffinement des intentions permet non seulement la description des services mais aussi l'amélioration de la performance dans la découverte de ces services.

Aucun de ces travaux [1][9] [11][14][19] n'inclut la notion de contexte, contrairement à Bonino et al. [5], qui présente un *framework* de découverte dynamique de services guidé par l'intention mais qui utilise aussi des informations sur le contexte d'utilisation. Cette approche est centrée autour de l'utilisation du contexte afin d'analyser les besoins exprimés par les utilisateurs. Pour cela, ces auteurs identifient à priori un ensemble d'intentions spécifiques à un domaine et les différentes tâches qui permettent leur accomplissement. Les services sont ainsi associés à ces tâches, permettant une découverte guidée par l'intention. Cependant, cette approche nous paraît trop restrictive car elle ne permet pas la satisfiabilité d'une intention qui n'était pas identifiée au préalable. Par ailleurs, la notion de contexte est utilisée uniquement comme paramètre d'entrée pour les services recherchés.

Même s'ils diffèrent, tous ces travaux défendent une évolution des SI vers des SI centrés utilisateur, soit par une approche intentionnelle, soit par une approche contextuelle. Nous pensons que ces approches sont en réalité complémentaires, et qu'une telle évolution ne peut être atteinte véritablement que par la combinaison de ces deux approches. À notre avis, seulement un mécanisme de découverte de services basé à la fois sur le contexte et sur l'intention de l'utilisateur est en mesure de répondre à des questions telles que “pourquoi un service est utile dans un contexte donné ?” ou “dans quelles circonstances émerge le besoin d'un service ?”. Dans notre approche, une intention est valable dans un contexte donné tout comme le contexte interfère sur la satisfiabilité de cette intention. À notre connaissance, aucun des travaux cités ne propose un mécanisme de découverte de services qui combine réellement le contexte et l'intention.

3 Découverte de Services Guidée par le Contexte et l'Intention

Afin d'accroître la transparence et l'efficacité des Systèmes d'Information Pervasifs, nous avons présenté dans [12][13] les bases d'une approche guidée par le contexte et l'intention. Nous soutenons que la compréhension de l'intention de l'utilisateur peut conduire à une meilleure compréhension de l'utilisation réelle des services, ce qui par conséquent peut améliorer la précision et la relevance des services choisis pour satisfaire les besoins des utilisateurs. Toute aussi importante, l'information contextuelle doit jouer un rôle central dans le mécanisme de sélection, car le contexte d'utilisation a un impact certain sur le choix du service qui est le mieux adapté. Ainsi, *l'intention* doit être utilisée afin d'exposer les besoins auxquels un service est censé répondre dans un contexte donné et d'ainsi implémenter une vision des SIP centrée sur l'utilisateur. Dans ce travail nous présentons un algorithme de correspondance sémantique proposé pour démontrer cette approche Intention/Contexte. Par la suite, nous analysons les premiers résultats expérimentaux issus d'une implémentation de cet algorithme.

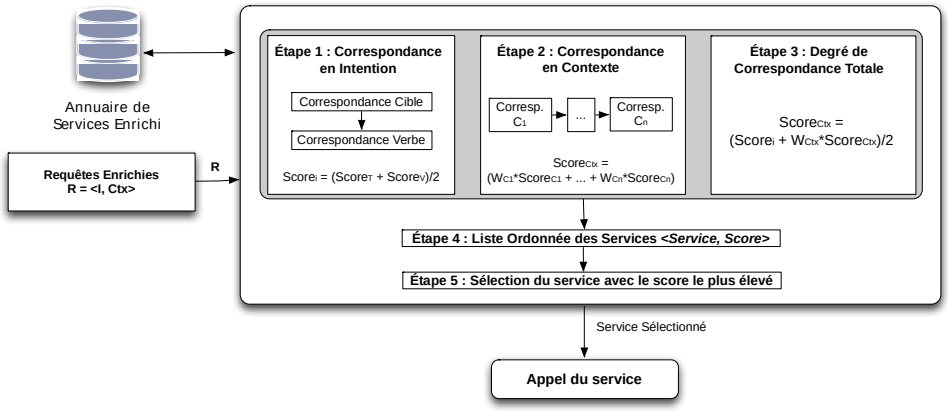

FIG. 1. Schéma du mécanisme de découverte de services guidé par l'Intention et le Contexte

Cet algorithme est décomposé en cinq étapes, comme indiqué schématiquement dans la FIG. 1. Tout d'abord, l'intention de l'utilisateur est comparée à l'intention associée à chaque service (i.e. les objectifs que le service doit satisfaire - étape 1). Ensuite, ce sont les conditions de contexte associées aux services qui sont comparées au contexte courant de l'utilisateur (étape 2). Finalement, le degré de correspondance global entre la requête de l'utilisateur (intention et contexte) et les services proposés est calculé par la somme des degrés de correspondance en intention et en contexte (étape 3), ce qui permet de classer les services selon leurs scores (étape 4). À partir de ce classement le service qui présente le score le plus élevé est proposé à l'utilisateur (étape 5).

3.1 Représentation Enrichie des Services

Lorsqu'un utilisateur fait appel à un service, il choisit en réalité l'intention que le service est supposé satisfaire. Cette intention émerge dans un contexte donné, ce qui caractérise également le service. Ainsi, nous avons enrichi la description de services OWL-S afin d'y inclure les informations relatives au contexte et à l'intention qui caractérisent un service [12]. Les informations liées à l'intention sont décrites grâce à l'addition d'une sous-ontologie qui représente l'intention qu'un service est censé satisfaire. Les ontologies définissant les intentions s'établissent dans de communautés d'experts partageant une même vision de leurs domaines respectifs, à l'instar des communautés soutenues par Mirbel et al. [11]. Cette vision s'adapte parfaitement aux SIP, puisque les services proposés dans ces systèmes le sont pour une communauté d'utilisateurs précise.

Par ailleurs, le contexte est décrit par une URL qui référencie un fichier externe, ce qui permet au fournisseur des services de mettre à jour facilement l'information de contexte liée à la description d'un service. Grâce à cette extension OWL-S, nous pouvons décrire l'intention qu'un service est censé satisfaire et les conditions de contexte dans lesquelles ce service est valable. Une explication détaillée de cette extension est présentée dans [12].

Lors de l'implémentation de notre mécanisme de découverte de services, nous avons utilisé l'API OWL-S Mindswap [15], laquelle a été enrichie par la description de l'intention et du contexte. Cette implémentation a été validée à l'aide du jeu de tests OWLS-TC2 [16], dont les services ont été modifiés afin d'inclure leurs descriptions intentionnelles et contextuelles. Le choix de ce jeu de tests est dû au grand nombre de services issus d'une large variété de domaines fournis.

3.2 Algorithme de Découverte de Services

L'algorithme de découverte de services proposé (résumé par la FIG. 2) utilise la description enrichie mentionnée ci-dessus pour procéder à une découverte guidée à la fois par le contexte et l'intention. L'utilisateur recherche, en fait, une intention. Sa requête est alors enrichie avec son contexte d'utilisation (voir [13] pour plus de détails). C'est cette requête comportant l'intention recherchée par l'utilisateur et son contexte courant que l'algorithme de découverte va comparer aux descriptions de service également enrichies. Pour tout service disponible, le score S_{score} est estimé à partir du degré de correspondance entre la requête de l'utilisateur et le service (lignes 6-11). Pour cela, le score I_{score} (degré de correspondance en intention) entre l'intention utilisateur - I_U - et l'intention du service - I_S - est calculé (ligne 7). L'intention est représentée par la somme de deux éléments obligatoires : le verbe (V) indiquant l'action principale et la cible (T) spécifiant l'objet sur lequel s'applique l'action [9][20]. Ainsi, le degré de correspondance entre l'intention de l'utilisateur et celle associée à un service est composé par : (***i***) le degré de correspondance entre la cible indiquée par l'utilisateur et celle indiquée par le service (respectivement T_U et T_S) ; et (***ii***) le degré de correspondance sémantique entre les verbes de l'utilisateur et celui associé à chaque service (V_U et V_S). Une fois que le score de correspondance en intention a été établi, c'est au tour du calcul du degré de correspondance par rapport au contexte (C_{score}). Pendant ce calcul, le contexte courant de l'utilisateur est comparé aux conditions contextuelles définies par la description du service (ligne 9). Les deux valeurs obtenues (I_{score} et C_{score}) sont utilisées pour le calcul du score final S_{score} (ligne 10). Les sections suivantes détaillent chaque étape de l'algorithme.

```
1: Procedure SERVICEDISCOVERY (C_U, I_U, S)
2: S_ranked = Ø
3: I_score = 0
4: C_score = 0
5: S_score = 0
6: for ∀s ∈ S do
7:     I_score = IntentionMatching(I_U, I_S)
8:     if I_score > k then
9:         C_score = ContextMatching(C_U, C_S)
10:        S_score = (I_score + (w_i × C_score))/2
11:    end if
12:    if S_score > l then
13:        S_ranked.add(s, S_score)
14:    end if
15: end for
16: return S_ranked
17: end procedure
```

FIG. 2. Algorithme pour la découverte de services

Correspondance en Intention

La correspondance intentionnelle (*intention matching*) est obtenue à partir de deux relations, *TargetMatch* et *VerbMatch*, qui sont utilisées pour définir la relation *IntentionMatch* entre l'intention de l'utilisateur $I_U = <V_U, T_U>$ et l'intention déclarée du service $I_S = <V_S, T_S>$: $\boldsymbol{IntentionMatch(I_U, I_S) = \forall V_U, \exists V_S : VerbMatch(V_U, V_S) \wedge \forall T_U, \exists T_S : TargetMatch(T_U, T_S)}$

La relation *TargetMatch* compare les concepts définis sur une ontologie de cible créée à partir des intentions. L'évaluation du niveau de correspondance entre les cibles T_U et T_S (utilisateur et service, respectivement) considère si un concept fourni par un service peut répondre à la cible demandée par l'utilisateur. Pour cela, nous avons utilisé un algorithme basé sur celui proposé par [17], qui met en relation un service demandé par rapport à un ensemble de services offerts. Dans [17], l'analyse des correspondances suit quatre niveaux distincts, présentés ci-dessous :

- **Exact** : le concept demandé correspond exactement au concept proposé ;
- **Plug-In** : le concept demandé est compris dans le concept proposé ;
- **Subsume :** le concept demandé comprend (*subsumes*) le concept proposé ;
- **Fail** : les deux concepts ne sont pas liés.

La correspondance des verbes *VerbMatch* est également basée sur une ontologie, laquelle contient un ensemble de verbes liés à un domaine, leurs significations et les relations entre ces verbes. Chaque relation relie un verbe à d'autres verbes voisins dont le significat est soit plus général, soit plus spécifique, mais aussi à des verbes similaires. Ainsi, l'évaluation de la correspondance entre les verbes peut se faire grâce aux cinq niveaux suivants :

- **Exact** : le verbe demandé est équivalent au verbe proposé ;
- **Synonym** : le verbe demandé partage une signification commune avec le verbe proposé ;
- **Hyponym** : il y a une relation de subordination entre le verbe demandé et le verbe proposé, qui a une signification plus générale ;
- **Hypernym** : il y a une relation de subordination entre le verbe demandé et le verbe proposé, qui a une signification plus spécifique ;
- **Fail** : aucune relation entre les verbes ne peut être établie.

Ces niveaux sont utilisés dans une relation *Property (P, $\mathcal{C}_d$, $\mathcal{C}_p$)*, où $\mathcal{C}_\mathrm{d}$ est le concept demandé, $\mathcal{C}_\mathrm{p}$ est le concept proposé et P indique la relation voulue entre $\mathcal{C}_\mathrm{d}$ et $\mathcal{C}_\mathrm{p}$ avec les niveaux décrits ci-dessus, i.e., $Property(P, \mathcal{C}_d, \mathcal{C}_p) = \forall \mathcal{C}_d, \mathcal{C}_p, \exists P: \mathcal{C}_d.P = \mathcal{C}_p$. Ainsi, une relation entre deux verbes "***Reserve***" et "***Book***" (très utilisés dans le domaine hôtelier) peut être représentée dans la base ontologique sous la forme ***Reserve.hasSynonym = Book***.

Chacune de ces relations correspond à un score allant de 0 (**Fail**) à 1 (**Exact**). Le score correspondant aux relations intermédiaires est calculé en fonction du nombre de niveaux qui séparent les concepts dans l'ontologie. Ainsi, si le score de la correspondance des cibles dépasse un seuil minimum alors on procède à la correspondance des verbes. Dans ce cas, le score final pour la correspondance en intention est défini par la somme des scores cible et verbe. Uniquement les correspondances dont le score dépasse un seuil minimum seront admises au niveau suivant pour l'analyse de la correspondance contextuelle (ligne 8, FIG. 2). Dans le cas échéant, le service en question n'est pas retenu et on passe au service suivant.

Correspondance en Contexte

La description du contexte pour un utilisateur (C_U) ou service (C_S) représente des ensembles non-vides d'éléments de contexte observables, i.e., $C_U = \{c_j\}$ pour $j > 0$ et $C_S = \{c_i\}$ pour $i > 0$. Chaque élément de contexte est décrit par une entité (à qui le contexte se réfère), un attribut (qui caractérise la propriété observée) et un ensemble de valeurs observées.

La relation *ContextMatch* dépend de la sous-relation *ContextElementMatch*, laquelle associe individuellement les différents éléments composant les descriptions de contexte C_U et C_S. *ContextMatch* est ainsi définie :

$$ContextMatch(C_S, C_U) = \forall c_i \in C_S, \exists c_j \in C_U : ContextElementMatch(c_i, c_j)$$

Les éléments de contexte c_i et c_j sont décrits par le triplet < *entity, attribut, value* > représentant l'entité observée, l'attribut observé et sa valeur. L'évaluation des ces éléments de contexte (*ContextElementMatch*) suit trois étapes :

i. Pour chaque pair d'éléments c_i et c_j, c_i.*entity* est comparé à c_j.*entity*, i.e., nous vérifions si les éléments de contexte peuvent être comparés ;
ii. Si le score obtenu dans l'étape précédente est supérieur à un seuil donné, la comparaison entre les attributs c_i.*attribut* et c_j.*attribut* est effectuée ;
iii. Si les attributs ont également une correspondance supérieure à un seuil donné, alors les valeurs c_i.*value* et c_j.*value* sont comparées

Comme les attributs de chaque élément de contexte peuvent avoir des valeurs aussi bien numériques que non-numériques (symboliques), l'évaluation de *ContextElementMatch* doit identifier la nature de ces valeurs et déclencher une fonction de comparaison adaptée aux types de données concernés. Ce type de fonction utilise des comparateurs génériques (égalité, différence, intervalle, supériorité, infériorité) afin d'indiquer si les éléments qui caractérisent le contexte

courant de l'utilisateur peuvent satisfaire ou non les conditions spécifiées par la description des services. Par exemple, si la définition de contexte d'un service indique que le débit du réseau doit être supérieur à 12500 bits/s et que la valeur observée par l'utilisateur est supérieure à cela, la fonction de comparaison retournera 1 (correspondance exacte) pour indiquer que le contexte de l'utilisateur satisfait la condition imposée par le service.

Après l'analyse des différents éléments de contexte, le score C_{score} est calculé par la somme pondérée des scores de chaque élément. Le poids associé à chaque attribut est défini par l'utilisateur et varie entre 0 et 1, représentant de cette manière l'importance de l'élément de contexte dans les préférences de cet utilisateur :

$$\boldsymbol{C_{score}} = \sum_{\boldsymbol{i=1,j=1}}^{\boldsymbol{n}} \boldsymbol{w_c} \times \boldsymbol{f}(\boldsymbol{c_i}, \boldsymbol{c_j})$$

Il convient d'observer que le contexte d'utilisation utilisé pour calculer le score ci-dessus correspond à celui observé lorsque l'utilisateur réalise sa requête. Ce contexte d'utilisation étant naturellement dynamique, une même intention exprimée par l'utilisateur peut être à l'origine de différentes requêtes enrichies, en fonction du contexte observé. Pour une même intention demandée, l'utilisateur pourra avoir ainsi des réponses différentes, adaptées à son contexte courant.

3.3 Implémentation

Le mécanisme de découverte de services présenté dans les sections précédentes a été implémenté en Java à partir de la plate-forme de services proposée dans [20]. À la base, celle-ci est organisée autour de trois interfaces (FIG. 3) : *ServiceManager*, *PersistenceManager* et *SearchEngine*. L'interface *ServiceManager* représente le point d'entrée de l'application et offre un ensemble de méthodes qui supportent la gestion des ontologies, la découverte de services et leur sélection. L'interface *PersistenceManager* agit comme une façade entre la *ServiceManager* et la base de services, ce qui permet l'accès aux descriptions des services. Finalement, l'interface *SearchEngine* est responsable par la recherche du service le plus approprié à une requête donnée.

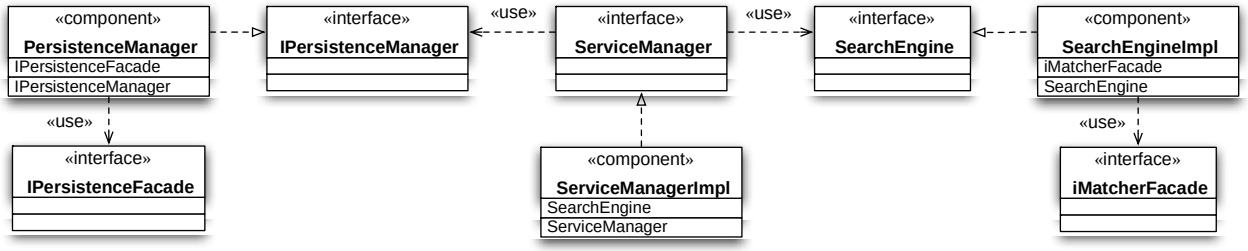

FIG. 3. Éléments du mécanisme de découverte sémantique de services [20]

Dans cette architecture, l'implémentation de l'interface *SearchEngine* (*SearchEngineImpl*) fournit les méthodes de découverte de services qui peuvent être utilisées par l'application. Chaque méthode doit être capable de proposer une liste de services ordonnée par leurs scores de correspondance. C'est à partir de ces scores que les méthodes de sélection de services choisissent le service qui s'adapte au mieux à la requête de l'utilisateur.

Afin de permettre l'évolution de l'API, cette première implémentation (*SearchEngineImpl*) utilise l'interface *IMatcherFacade,* laquelle offre une interface générale pour l'implémentation de nouvelles méthodes de découverte de services (appelées *matchmaker*). La sélection des *matchmakers* à déployer se fait ainsi par la simple édition d'un fichier de propriétés.

Dans nos expériences, deux implémentations de *matchmaker*, correspondant à deux mécanismes de découverte de services distincts ont été proposées. La première implémentation, appelée *BasicMatchmaker* [20], utilise uniquement les références aux informations d'entrée (*input*) et de sortie (*output*) fournies par l'utilisateur pour procéder à la sélection des services. Cette implémentation agit comme implémentation de référence dans nos évaluations, puisqu'elle adopte

une approche traditionnelle basée sur les spécifications fonctionnelles d'un service : l'utilisateur est obligé, dans ce cas, de spécifier les entrées et les sorties qu'il attend du service recherché, au lieu de se concentrer uniquement sur l'intention qu'il souhaite satisfaire.

Pour la deuxième implémentation, appelée *ContextIntentionMatchmaker*, nous mettons en œuvre notre mécanisme de découverte de services avec prise en charge du contexte et de l'intention. Pour cela, *ContextIntentionMatchmaker* utilise une API étendue de OWL-S [15], le *framework* Jena [8] et le moteur de raisonnement Pellet [18]. Pour chaque service dans la base de connaissances, il est calculé un score basé sur l'intention et sur le contexte de l'utilisateur, par rapport aux définitions des services disponibles. Ce mécanisme est composé de deux classes *ContextMatching* et *IntentionMatching*, nécessaires au calcul des scores en contexte et en intention, respectivement. La séparation des deux éléments permet l'évaluation de chaque composant et l'analyse de leur impact sur le mécanisme de sélection. La section suivante analyse les résultats des expérimentations qui ont été menées afin de comparer le mécanisme que nous proposons ici à une approche plus traditionnelle, représentée par *BasicMatchMaker*.

4 Évaluation

L'évaluation des différentes méthodes de découverte de services a été effectuée sur une base sémantique contenant un ensemble étendu de descriptions de services, issu de la base OWLS-TC2. Parmi les domaines disponibles, nous avons choisi les services relevant le domaine du tourisme. Ce domaine contient autour de 200 descriptions de service, qui ont été enrichis avec des informations contextuelles et intentionnelles.

Dans un soucis d'analyse des performances, nous avons pu effectuer des mesures sur les équipements appartenant à la plate-forme Grid'5000 [7]. Grid'5000 est une grille de calcul expérimentale distribuée sur plusieurs institutions en France et à l'étranger. Plus qu'un équipement dédié au calcul de haute performance (HPC), la grille Grid'5000 est aussi un réseau atypique composé de grappes de calcul hétérogènes, comme illustré par le TAB. 1. C'est cette hétérogénéité qui nous conduit à utiliser Grid'5000, car à travers cette grille nous avons pu évaluer notre implémentation sur de multiples architectures, ce qui nous a permis d'avoir une vue globale de son comportement. Ainsi, les sections suivantes décrivent les résultats obtenus sur cette grille.

Cluster (année)	CPU	Processeur	Cœurs	RAM (Go)
Stremi (2011)	2	AMD 1.7 GHz	12	48
Graphene (2010)	1	Intel 2.53 GHz	4	16
Griffon (2009)	2	Intel 2.5 GHz	4	16
Capricorne (2006)	2	AMD 2 GHz	1	2
Bordeplage (2007)	2	Intel 3 GHz	1	2
Bordereau (2007)	2	AMD 2.6 GHz	2	4
Borderline (2007)	4	AMD 2.6 GHz	2	32

TAB. 1. Caractéristiques des serveurs utilisés

L'ensemble des mesures que nous avons réalisé a pour objectif d'évaluer la validité des algorithmes proposés et l'adéquation des descripteurs étendus de service qui ont été implémentés. Deux observations principales dégagent de ces expériences :

- Scalabilité – analyse de l'impact du nombre de services disponibles sur le temps d'exécution des algorithmes de découverte de services ;
- Qualité du résultat – évaluation de l'efficacité des différents algorithmes vis-à-vis de leur capacité à trouver un service adéquat au contexte et à l'intention de l'utilisateur.

La première de ces observations, la scalabilité, nous permet d'analyser la faisabilité du mécanisme proposé. Celui-ci doit être capable de monter à l'échelle, son temps d'exécution doit

rester limité malgré un nombre grandissant de services analysés, pour ne pas gêner l'expérience de l'utilisateur.

La seconde observation, la qualité, considère deux métriques majeures dans le domaine des services : la précision et le rappel. Ces métriques nous permettent d'analyser si le mécanisme propose atteint réellement son objectif, puisqu'il est censé fournir des résultats mieux adaptés à l'utilisateur par rapport à une approche traditionnelle.

4.1 Scalabilité

La scalabilité est la capacité d'un système à s'adapter au rythme de la demande [4]. Souvent ignorée, l'analyse de cette propriété est un facteur essentiel lors du développement des systèmes répartis car cette propriété permet de déceler les possibles limitations du système lors d'une montée en charge. Cette montée en charge peut comprendre autant le nombre de clients (utilisateurs) que le nombre d'éléments dans la base de connaissances. Dans le cas des mécanismes de découverte de services, la scalabilité a un impact direct sur le temps d'évaluation des requêtes, ce qui peut gêner l'expérience de l'utilisateur si ce temps devient trop important.

Dans les expériences que nous avons menées, la scalabilité des mécanismes de découverte de services a été représentée par le temps moyen de traitement, lorsqu'on varie le nombre de services disponibles dans la base de connaissances. L'utilisation des ressources de Grid'5000 nous a permis aussi de comparer la performance des algorithmes lorsqu'ils sont déployés sur des machines avec des caractéristiques différentes, comme illustré par la FIG. 4(a).

Le temps de traitement des requêtes a été mesuré en faisant varier le nombre de services entre 10 et 200 services. Les résultats démontrent que le temps de réponse de l'algorithme de découverte suit une tendance $y = ax^b$, ce qui nous permet d'affirmer que le mécanisme implémenté a une bonne scalabilité. Les variations les plus importantes sont observées lorsqu'on compare les différentes familles de processeurs (Intel et AMD) et leurs générations. Le nombre de cœurs n'a pas suffisamment influencé les résultats du fait que le prototype implémenté n'a pas été conçu pour le support aux architectures multi-cœur. Nous sommes certains que l'exploration parallèle de la base de services permettra de baisser considérablement le temps moyen de réponse, tout en gardant la scalabilité observée dans cette expérience. Il faut noter que la FIG. 4(a) présente seulement les courbes de performance pour l'algorithme Intention/Contexte pour des raisons de simplicité, car les observations ci-dessus sont aussi valables pour les autres algorithmes (voir FIG. 4(b)).

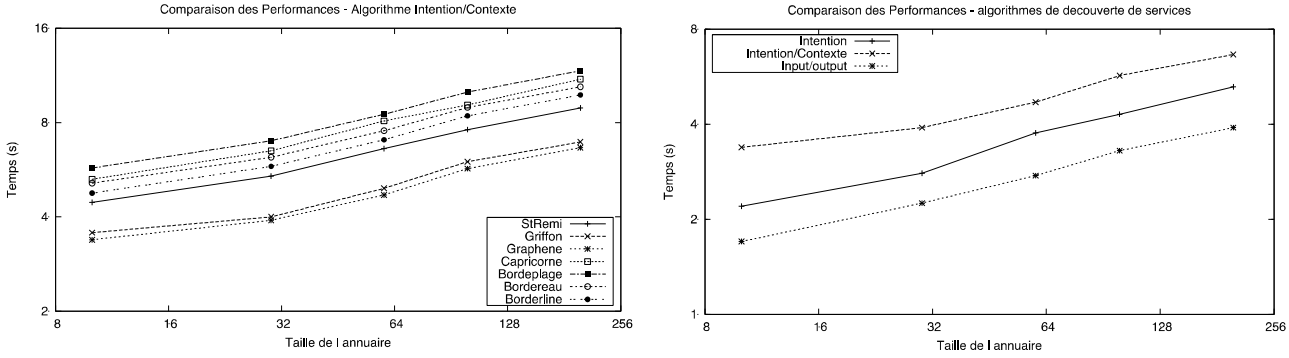

*FIG. 4. Comparaison de performances **(a)** entre serveurs et **(b)** entre différents algorithmes de découverte de services*

La FIG. 4(b) présente en détail la comparaison entre les trois algorithmes de recherche de services : (*i*) un mécanisme **Input/Output** (implémenté par la classe *BasicMatchFacade*) ; (*ii*) un mécanisme de découverte purement **Intentionnel** (implémenté par la classe *ContextIntentionMatchmaker*, dans laquelle la recherche guidée par le contexte est désactivée) ; et (*iii*) le mécanisme de découverte de services guidé par l'**Intention et le Contexte** implémenté par la classe *ContextIntentionMatchmker*. Ces trois algorithmes, appelés respectivement IO, I et IC, sont représentés par leurs performances sur la grappe *Graphene*. Cette grappe a été choisie par ses caractéristiques (nombre de CPUs et de cœurs, quantité de mémoire, etc.), qui sont les plus proches de celles qu'on retrouve sur le marché. Les autres grappes, au contraire, sont soit trop

représentatives des machines pour le HPC (quantité importante de mémoire, grand nombre de cœurs) ou ont des configurations obsolètes dues à leurs générations.

Il convient également d'observer que la variation purement contextuelle de la classe *ContextIntentionMatchmaker* n'a pas été considérée car l'intérêt de l'usage du contexte dans la découverte de services a déjà été démontré par des nombreux travaux [3][21].

Dans la FIG. 4(b) nous observons que le mécanisme de découverte guidé par le contexte et l'intention (IC) a un temps moyen de réponse plus élevé que les deux autres algorithmes. Cette différence, surtout lorsqu'elle est comparée à l'algorithme input/output (IO), n'est pas inquiétante et reste raisonnable. Ceci est une donnée importante pour nous, car elle démontre la faisabilité de notre proposition sur des machines habituellement utilisées comme annuaire de service et ceci malgré une implémentation non optimisée aux architectures multi-cœur. De plus, nous devons considérer que l'algorithme IO utilise un mécanisme très simplifié qui a un coût réduit mais, comme le démontre la prochaine section, il présente une qualité de résultat bien inférieure à celle de l'algorithme Intention/Contexte (IC).

4.2 Qualité des Résultats

Afin d'évaluer la qualité des résultats, nous avons mesuré deux des principales métriques de qualité dans la sélection de services : la *précision* et le *rappel*. Ces deux métriques sont définies par deux ensembles, l'ensemble des items trouvés et l'ensemble des items pertinents. La métrique *précision* indique la capacité d'un système à retrouver uniquement les items pertinents (i.e., sans faux-positifs), alors que la métrique *rappel* mesure la capacité d'un système à retrouver tous les services pertinents [24]. Ces deux métriques sont définies comme suit :

$$Précision = \frac{|\{items\ pertinents\} \cap \{items\ retrouvés\}|}{\{items\ retrouvés\}} \quad Rappel = \frac{|\{items\ pertinents\} \cap \{items\ retrouvés\}|}{\{items\ pertinents\}}$$

Afin d'évaluer ces deux métriques, nous avons formulé cinq requêtes différentes relatives au domaine du tourisme et qui doivent retrouver les services correspondants dans la base de connaissances. Ces requêtes sont caractérisées par l'intention de l'utilisateur et son contexte actuel, comme indiqué au TAB. 2. Dans ce scénario, un utilisateur souhaite pratiquer un sport pendant les vacances. Cet utilisateur est intéressé par le surf (*surfing*) ou la randonnée (*hiking*). Ainsi, il recherche des destinations où ces sports peuvent être pratiqués. Par la suite, il souhaite réserver une chambre dans un hôtel ou *Bed-and-Breakfast* pendant cette période.

Intention	Context
Reserve Hotel	- Age >=18
Reserve BedAndBreakfast	- Age >= 18 - Season = Summer
Locate Sport Destination	-Age >= 18 - Season = Summer - City = Germany
Search Surfing Destination	-Age >= 18 -Surfing-Level=Beginner -Season=Summer -Weather=notDisturbed
Search Hiking	-Age >= 18 - Hiking Level=Confirmed -Weather=not disturbed -Health = Good

TAB. 2. Intention de l'utilisateur dans un contexte donné

À partir des résultats obtenus sur la grappe *Graphene*, nous observons les métriques *précision* et *rappel* pour les différents algorithmes. Les résultats présentés en FIG. 5(a) indiquent que l'algorithme guidé par l'intention et le contexte (IC) présente un niveau de **précision** très supérieur à celui obtenu par l'algorithme de base (IO). La précision moyenne de l'algorithme IC est proche à 99%, alors que celle de l'algorithme IO est proche à 50%. Ces résultats indiquent que le mécanise de découverte de services guidé par l'intention et le contexte a une plus grande chance de retrouver le service le plus approprié pour l'utilisateur.

Les bons résultats obtenus pour la *précision* sont accompagnés par d'intéressants résultats concernant le *rappel*, comme l'illustre la FIG. 5(b). Effectivement, le taux de *rappel* moyen pour l'algorithme IC avoisine les 95%. L'algorithme IO a aussi un taux moyen de 93.2%, mais ces résultats sont circonstanciels vu que cet algorithme n'est pas en mesure de choisir un service qui

s'adapte au contexte de l'utilisateur ni à son intention. En effet, l'algorithme IO se limite à retourner l'ensemble de services qui peuvent concerner la requête, avec un taux élevé de faux-positifs (indiqué par sa *précision*).

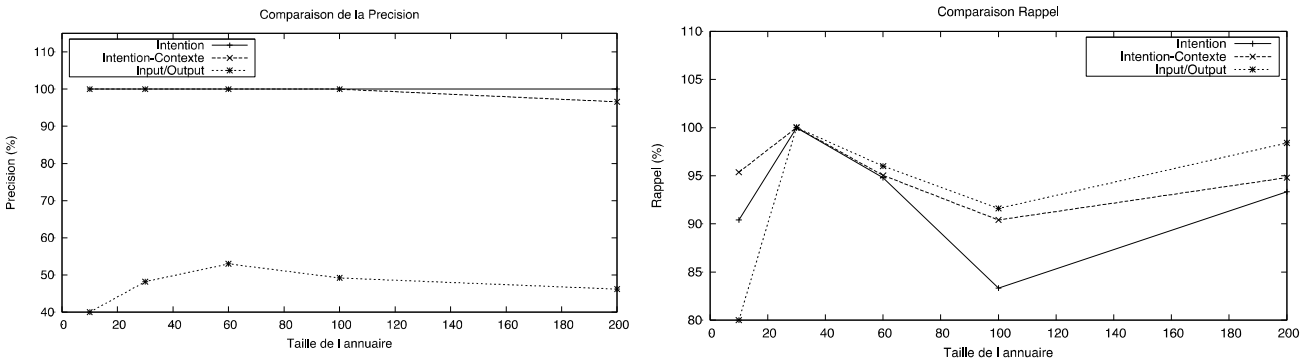

*FIG. 5. Comparaison des métriques **(a)** Précision et **(b)** Rappel*

Contrairement à ce qui est observé avec l'algorithme IO, l'analyse conjointe des métriques *précision* et *rappel* démontre que l'algorithme IC réussit à trouver tous (ou presque tous) les services qui correspondent à l'intention et au contexte de l'utilisateurs, et cela avec le plus faible taux de faux-positifs. Ces propriétés font partie des caractéristiques qui contribuent à la qualité de l'expérience de l'utilisateur.

L'analyse de ces résultats démontre ainsi l'intérêt de l'approche proposée ici. Nous pensons que le mécanisme de découverte proposé permet réellement de sélectionner le service qui correspond au mieux aux besoins de l'utilisateur, et ceci grâce à la fois à son approche intentionnelle, laquelle est plus transparente pour l'utilisateur, et à l'usage du contexte qui limite les services à ceux qui sont valides, i.e., dont l'intention émerge dans un contexte compatible.

5 Conclusions

Avec la démocratisation des dispositifs et des réseaux mobiles, les systèmes d'information deviennent pervasifs. Malheureusement, l'accès aux services dans un tel environnement reste trop complexe, car souvent l'utilisateur doit choisir seul un service parmi plusieurs implémentations disponibles, sans avoir pour autant le bagage nécessaire pour comprendre ces implémentations. Afin d'augmenter la transparence de ces systèmes, nous devons offrir à l'utilisateur le service qui satisfait ses besoins, sans qu'il soit forcé de connaître les détails sur l'implémentation et ses contraintes.

Dans cet article, nous proposons une approche centrée sur l'utilisateur capable d'apporter des services adaptés au contexte d'utilisation tout en gardant un niveau de transparence convenable.

Afin de démontrer cette approche, nous avons implémenté un mécanisme de découverte de services guidé non seulement par le contexte d'utilisation, mais également par son intention. Ce mécanisme de découverte est ainsi évalué sous deux facettes : la scalabilité, qui nous permet d'analyser la faisabilité du mécanisme proposé notamment par rapport à la montée en charge des annuaires de service ; et la qualité des résultats, dans laquelle deux métriques majeures, la précision et le rappel, permettent d'analyser si le mécanisme proposé atteint réellement son objectif. Des mesures effectuées sur un ensemble très diversifié de machines appartenant à la plate-forme Grid'5000 nous permettent de démontrer la faisabilité de notre proposition. Les résultats obtenus démontrent que notre algorithme est capable de sélectionner, en un temps de réponse raisonnable, tous (ou presque tous) les services qui répondent à l'intention de l'utilisateur dans son contexte d'usage (avec un taux très faible de faux-positifs). Ainsi, ceci nous incite à poursuivre les travaux en cours.

Les prochaines étapes de notre travail doivent se concentrer d'une part sur l'optimisation de l'implémentation vis-à-vis du parallélisme intrinsèque des processeurs actuels, et d'autre part, sur la prédiction de services, i.e., la possibilité d'anticiper les intentions futures de l'utilisateur. De plus,

nous envisageons de tester notre approche avec un scénario plus complexe et de l'évaluer avec une base sémantique contenant un ensemble plus large de descriptions de services.

6 Références

[1] Aljoumaa, K., Assar, S. and Souveyet, C. *Reformulating User's Queries for Intentional Services Discovery Using an Ontology-Based Approach*, 4th IFIP Int. Conf on New Technologies, Mobility and Security (NTMS), Paris, France, pp. 1-4, 2011.

[2] Bell G. and Dourish, P. *Yesterday's tomorrows: notes on ubiquitous computing's dominant vision*, Personal and Ubiquitous Computing, 11(2), Springer London, pp. 133-143, 2007.

[3] Ben Mokhtar, S., Preuveneers, D., Georgantas, N., Issarny, V. and Berbers, Y. *EASY: Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support*, Journal of Systems and Software 81(5), pp.785-808, 2008.

[4] Bondi, A. B. *Characteristics of scalability and their impact on performance*, 2nd International Workshop on Software and Performance (WOSP), pp. 159-203, 2000.

[5] Bonino da Silva Santos, L.O., Guizzardi, G., Pires, L.F. and Van Sinderen, M. *From User Goals to Service Discovery and Composition*, ER Workshops, pp. 265-274, 2009.

[6] Dey, A. *Understanding and using context*, Personal and Ubiquitous Computing, 5(1), pp. 4-7, 2001.

[7] Grid5000, https://www.grid5000.fr/, 2012.

[8] Jena Semantic Web Framework, http://jena.sourceforge.net/.

[9] Kaabi, R.S. and Souveyet, C. *Capturing intentional services with business process maps*, 1st Int. Conference on Research Challenges in Information Science (RCIS 2007), pp. 309-318, 2007.

[10] Kouruthanassis, P.E. and Giaglis, G.M. *A design theory for pervasive information systems*, 3rd Int. Workshop on Ubiquitous Computing (IWUC'06), pp. 62-70, 2006.

[11] Mirbel, I. and Crescenzo, P. *From end-user's requirements to Web services retrieval: a semantic and intention-driven approach*, J.-H. Morin, J. Ralyte, M. Snene, "Exploring service science," First Int Conf, IESS, Springer, pp. 30-44, 2010.

[12] Najar, S., Kirsch-Pinheiro, M. and Souveyet, C. *The influence of context on intentional service*, 5th Int. IEEE Workshop on Requirements Engineerings for Services (REFS'11) - IEEE Conference on Computers, Software, and Applications (COMPSAC'11), Munich, Germany, pp. 470 – 475, 2011.

[13] Najar, S., Kirsch-Pinheiro, M., and Souveyet, C., *Towards semantic modeling of intentional pervasive information systems*, 6th Workshop on Emerging Web Services Technology (WEWST 2011), 9th European Conference on Web Services (ECOWS 2011), pp. 30-34, 2011

[14] Olsson, T., Chong, M.Y., Bjurling, B. and Ohlman, B. *Goal Refinement for Automated Service Discovery*, 3rd Int. Conf on Advanced Service Computing, Service Computation'11, Rome, Italy, pp. 46-51, 2011.

[15] OWL-S API: http://www.mindswap.org/2004/owl-s/api/

[16] OWLS-TC: http://www.semwebcentral.org/projects/owls-tc/

[17] Paolucci, M., Kawmura, T., Payne, T. and Sycara, K. *Semantic matching of web services capabilities*, in: Proceedings of the 1st Int. Semantic Web Conference, Springer LNCS 2342, 2002.

[18] Pellet, http://www.mindswap.org/2003/pellet/index.shtml

[19] Rolland, C., Kirsch-Pinheiro, M., Souveyet, C. *An Intentional Approach to Service Engineering*, IEEE Transactions on Service Computing, Vol.3 n°4, pp. 292-305, 2010.

[20] Schulthess, S. *Construction of a registry for searching web service*, Master Thesis, EFREI, Engineering School Paris, 2011.

[21] Toninelli, A., Corradi, A. and Montanari, R. *Semantic-based discovery to support mobile context-aware service access*, Computer Communications, vol.31 n° 5, pp. 935-949, 2008.

[22] Vanrompay, Y., Kirsch-Pinheiro, M., Berbers, Y., *Service Selection with Uncertain Context Information*, In: Stephan Reiff-Marganiec and Marcel Tilly (Eds.), Handbook of Research on Service-Oriented Systems and Non-Functional Properties: Future Directions, IGI Global, pp. 192-215, 2011.

[23] Weiser, M. *The computer for the 21st Century*, Sci Am 265(3), pp. 94–104, 1991.

[24] Xiao, H., Zou, Y., Ng, J. and Nigul, L. *An Approach for Context-aware Service Discovery and Recommendation*, IEEE Int. Conf on Web Services (ICWS), Miami, pp.163-170, 2010.