DAML-S : interactions, critique et évaluation.

Gautier Dallons

Institut d'informatique des FUNDP
Namur, Belgique

Résumé. Internet constitue un réservoir de ressources. Ces ressources présentées sous le nom de "Web Services" devraient pouvoir être facilement exploitables de manière automatique par un agent autonome. Pour cela, diverses initiatives ont vu le jour telles que WSDL et UDDI afin de rendre cette exploitation possible. A cette fin, un nouveau langage de description sémantique de services web est proposé : DAML-S. DAML-S est une ontologie permettant de décrire les services web. Grâce à son aide, des tâches telles que la découverte, l'invocation, la composition et le suivi d'avancement de services web devraient pouvoir être gérés de manière automatique. Il devient donc intéressant de voir comment DAML-S interagit avec WSDL et UDDI, mais aussi de l'évaluer par rapport aux objectifs qu'il est sensé atteindre. Cet article mettra en évidence les interactions possibles entre DAML-S, WSDL et UDDI, pour ensuite proposer une critique et une évaluation de DAML-S.

1 Introduction

La vision par laquelle on percevait le Web a évolué. Ainsi, nous sommes passés d'une vision dans laquelle il était constitué d'un ensemble de pages accessibles par mots clefs à une vision du Web où celui-ci devient un fournisseur de ressources accessibles par leurs contenus.

La majorité des ressources du Web sont constituées par des appareils, des programmes et des contenus accessibles en ligne. Ainsi, la notion de "Web Service" est née. Un "Web Service" peut être considéré comme un site web qui ne propose pas seulement de l'information, mais qui autorise éventuellement d'effectuer une action ou un changement dans le "Monde" [1].

Ces "Web Services" représentent un défi informatique mais aussi économique. En effet, ils peuvent constituer un apport de rapidité et d'efficacité pour l'e-business. De même, ils permettent la mise en place de sites qui échangent dynamiquement de l'information. Ces sites peuvent servir de support à une nouvelle forme d'organisation dans laquelle les chaînes de produits et celles de livraison deviennent dynamiques et permettent d'augmenter ainsi la valeur ajoutée [2].

L'exploitation de ces ressources pose deux grands problèmes : il faut pouvoir les localiser et interagir avec elles pour composer des services plus complexes. La solution à ces problèmes demande un langage de description sémantique du contenu du web qui soit interprétable par un logiciel d'ordinateurs afin de rendre ces tâches, soit semi-automatiques, soit automatiques [3].

C'est dans ce contexte que DAML-S ("DARPA Agent Markup Language Service ontology") fut développé. Il se base sur le langage DAML+OIL qui permet de décrire une ontologie pour un domaine particulier. Une ontologie peut être vue comme une librairie qui définit un ensemble de termes communs, de concepts propres à un domaine, partagés par les personnes de ce domaine de telle sorte que la sémantique de ces concepts soit commune à ces personnes [4]. Ainsi, DAML-S propose une ontologie pour les "Web Services". Il permet de décrire ce que peut faire un service, comment l'utiliser et quels sont ses effets sur le "Monde".

DAML-S sert de support à nombreuses tâches :

1. la découverte et la sélection d'un service qui consistent à mettre en correspondance un demandeur de service avec un fournisseur qui répond à ce service.
2. l'invocation de ce service qui permet de réaliser l'appel effectif du service découvert.
3. l'interopération et la composition de services qui autorisent de composer des services plus simples afin d'obtenir un service complexe qui répond aux attentes d'un demandeur.
4. la surveillance de l'évolution du processus ("Monitoring") qui sert à surveiller l'état d'accomplissement et de déroulement du service.

Dans cet article, je présenterai les principales parties de DAML-S de manière conceptuelle sans entrer dans des détails pratiques ou syntaxiques. Lors de la présentation de chaque partie, j'identifierai la(les) tâche(s) supportée(s) par cette partie. Ensuite, je verrai les interactions possibles entre DAML-S et UDDI ("Universal Description Discovery and Integration") ainsi que celles avec WSDL ("Web Service Description Language"). Suite à cela, je critiquerai DAML-S selon les tâches qu'il permet de résoudre et selon les parties encore indéfinies de DAML-S. En examinant les conséquences de ces critiques, j'évaluerai DAML-S en termes d'utilité, d'atteinte de ses objectifs et en termes de facilité de mise en oeuvre. Enfin, je conclurai et j'émettrai de nouvelles perspectives pour OWL-S, le successeur de DAML-S.

2 Présentation de DAML-S

Cette présentation est une synthèse construite principalement à partir des articles [1,3]. DAML+OIL permet de définir les catégories abstraites des entités, événements, etc, en termes de classes et de propriétés. DAML-S utilise ce langage de description d'ontologie pour définir une ontologie particulière, celle des "Web Services". Cette ontologie permet de décrire les propriétés d'un "Web Service" ainsi que son ou ses services rendus disponibles au "Monde".

Les classes principales de l'ontologie décrites par DAML-S sont illustrées à la figure 1. Au coeur de cette ontologie, on trouve la classe service. Elle reprend les propriétés générales d'un "Web Service". Cette classe présente un "ServiceProfile", est décrite par un "ServiceModel" et supporte un "ServiceGrounding". Le

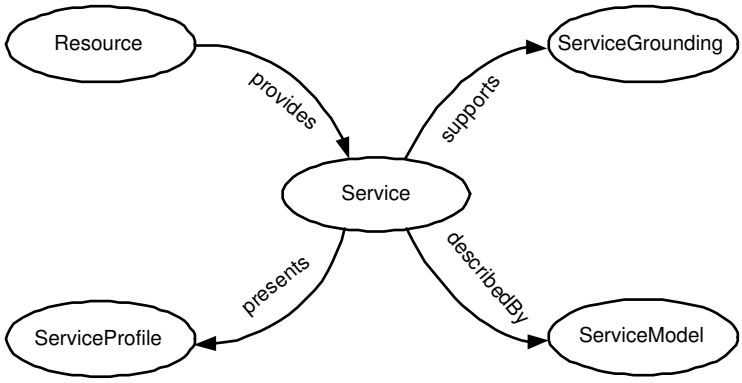

Fig. 1. Classes principales de l'ontologie DAML-S [3].

"ServiceProfile" explique ce que fait le service et ce qu'il exige des autres agents, le "ServiceModel" définit le fonctionnement du "Web Service", le "ServiceGrounding" fournit les informations nécessaires à l'utilisation du "Web Service" et la classe ressource donne des informations relatives aux ressources utilisées par le "Web Service". Ces différentes classes sont exposées dans les points suivants.

2.1 Le "ServiceProfile"

Le "ServiceProfile" décrit le service en fonction de ce qu'il fait afin de permettre à un demandeur de voir si le service proposé lui convient. Dès lors, le "ServiceProfile" propose une vue du "Web Service" décomposée en trois grands aspects :

1. une description du service et de son fournisseur. Ainsi, il présentera le nom du service, un texte descriptif, etc. De même, il décrira son fournisseur en termes de nom, d'adresse physique, d'adresse web, etc.
2. une description du comportement fonctionnel du service. Cette description est réalisée à l'aide d'un ensemble d'entrées du service, les "inputs", d'un ensemble de sorties du service, les "outputs", d'un ensemble de pré-conditions nécessaires au bon déroulement du service et un ensemble d'effets du service sur le "Monde".
3. une description des attributs fonctionnels utiles à la sélection automatique d'un "Web Service". Ainsi, on pourra trouver comme attributs fonctionnels, le temps de réponse, le coût du service, etc.

Le "ServiceProfile" sert de support à la découverte de "Web Services" et à leur sélection. Cette activité est appelée le "matching". Elle consiste, à partir d'un "ServiceProfile" hypothétique du service désiré par un demandeur, à fournir le "Web Service" du fournisseur qui propose un "ServiceProfile" qui correspond

le plus à la description du demandeur. Elle suppose l'intervention d'un agent intermédiaire de type annuaire de services qui implémente cet algorithme de "matching". L'algorithme de "matching" n'est pas défini à priori, c'est pourquoi certains articles proposent des solutions diverses [2,5]. La liberté d'implémentation permet de définir des politiques différentes de "matching" comme, par exemple, la répartition de charge au sein de services semblables ("Load balancing"). L'activité de "matching" suppose la connaissance de la sémantique du domaine dans lequel opèrent les demandeurs et les fournisseurs afin de pouvoir se dérouler correctement.

2.2 Le "ServiceModel"

Le "ServiceModel" sert à expliquer comment le service fonctionne. Un "Web Service" peut être vu comme un processus. C'est pourquoi, le "ServiceModel" possède une sous-classe : le "ProcessModel". Ce dernier comprend deux aspects. Le premier est le processus qui est décrit par l'ontologie de processus ("Process Ontology"). Le second aspect fait référence au modèle de contrôle du processus défini par l'ontologie de contrôle de processus ("Process Control Ontology").

L'ontologie de processus permet de décrire le processus en termes d'entrées ("inputs"), de sorties ("outputs"), de pré-conditions ("preconditions") et d'effets sur le "Monde" ("effects"). Cette ontologie définit trois sous-types de processus :

1. le processus atomique qui peut directement être invoqué par le demandeur. Il n'est pas composé de sous-processus. Le demandeur voit son exécution comme une opération primitive, c'est-à-dire qu'il voit l'exécution du processus en une seule étape.
2. le processus simple qui ne peut pas être directement invoqué par le demandeur. Il est aussi perçu par le demandeur comme étant constitué d'une étape unique. Son rôle est de permettre un certain niveau d'abstraction. Ainsi, on pourra, par exemple, proposer au demandeur une vue simplifiée d'un processus composé.
3. le processus composite qui est composé d'autres processus selon des séquences d'enchaînement. Ainsi, on trouvera, par exemple, comme séquences d'enchaînement, le "IF-THEN-ELSE", le "SPLIT", le "REPEAT-UNTIL", etc.

L'ontologie de contrôle de processus sert à définir un modèle de contrôle des instanciations d'un "Web Service" qui doit permettre à un agent de pouvoir surveiller l'état d'avancement du service et de contrôler le service. Pour cela, cette ontologie, qui n'est pas encore définie, doit procurer des règles de correspondance entre, d'un côté, l'état du processus avant l'invocation, défini par les entrées et les pré-conditions, et, de l'autre côté, l'état du processus après l'invocation. De plus, elle doit pouvoir décrire les contraintes temporelles et d'état engendrées par les séquences d'enchaînement. Enfin, elle doit donner une représentation de messages sur l'exécution des processus afin de pouvoir surveiller l'état d'avancement de ceux-ci.

Le "ServiceModel" permet de réaliser plusieurs tâches importantes sur les "Web Services". En effet, après la découverte d'un service grâce à son "ServiceProfile", le "ServiceModel" permet d'effectuer une analyse plus détaillée par un agent du service afin de déterminer si le processus répond véritablement aux attentes de cet agent. De plus, le "ServiceModel" sert de support à la composition de services complexes [6,7]. Il permet aussi de coordonner les différentes activités des services web. Enfin, il offre la possibilité de surveiller l'état d'avancement d'un service.

2.3 Le "ServiceGrounding"

Le "ServiceGrounding" établit une correspondance entre une spécification abstraite définie en DAML-S par les "IOPE's" ("Inputs Outputs Preconditions Effects") pour un "ServiceModel" et une spécification concrète. Cette mise en correspondance permet d'accéder au service. Pour cela, plusieurs aspects doivent être présents dans une description d'un "grounding". En effet, il faudra choisir un protocole à utiliser pour accéder au service, un format de messages, la façon de les sérialiser, quels mécanismes de transport utiliser et quel mode d'adressage employer. Nous voyons directement que ces points peuvent être réalisés grâce à un langage de description de services web : WSDL. Nous aborderons ce point plus en détail dans la section 4. Le "ServiceGrounding" sert donc de support à l'invocation du service.

2.4 Les ressources

Les "Web Services" ont souvent besoin de ressources pour pouvoir s'exécuter. Ces ressources sont variées et nombreuses. Il est donc intéressant de les définir dans une ontologie. Cette ontologie doit permettre un niveau d'abstraction assez élevé pour couvrir différentes ressources telles que les ressources temporelles, physiques, etc.

On peut distinguer deux grands types de ressources appelés "Allocation Types" : celles qui sont consommables ("ConsumableAllocation") et celles qui restent réutilisables ("ReusableAllocation"). Les premières disparaissent avec l'exécution du service, tandis que les secondes sont à nouveau disponibles après l'exécution du service. Remarquons que les ressources consommables peuvent être, après utilisation, réapprovisionnées.

La mesure de la quantité ("Capacity Types") d'une ressource peut être de deux sortes. La première est discrète ("DiscreteCapacity"), c'est-à-dire qu'une ressource de ce type est consommée en unités entières. La seconde est continue ("ContinuousCapacity").

Les ressources proposées par l'ontologie de ressources peuvent être atomiques ou agrégées. Dans le premier cas, nous avons une ressource unique qui est allouée pour l'utilisation du service. Dans le second cas, un ensemble ou un sous-ensemble de ressources doit être alloué afin d'exécuter le service. Nous n'entrerons pas davantage dans les détails de l'ontologie de ressources, car elle est en cours de développement et donc sujette à modification.

3 Interaction avec UDDI

UDDI permet la construction d'un registre de services web. Il ne propose aucune description du service sur base de ce qu'il offre, mais présente uniquement un nom, un pointeur (moyen d'accéder au service, port, etc) vers ce service et des attributs additionnels définis dans les "TModels" [1,8]. Ses capacités de recherche sont limitées car elles reposent sur une investigation par mots clefs sur base du nom pour les services et les "TModels". Cela ne permet pas un "matching" flexible tel que le permet DAML-S. En combinant UDDI et DAML-S, on peut arriver à résoudre ce problème de "matching" flexible. Cette solution, proposée dans l'article [2], est illustrée à la figure 2.

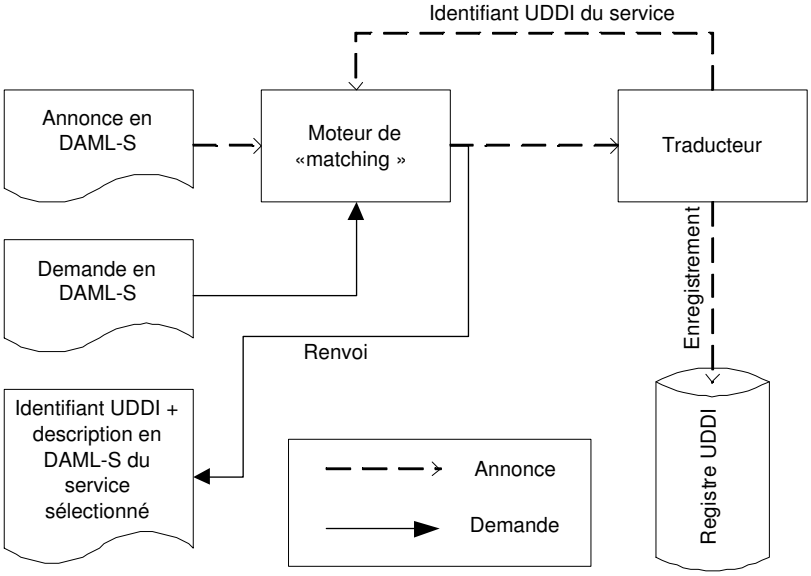

Fig. 2. Fonctionnement d'un moteur de "matching" utilisant UDDI.

Lors de la publication d'un service par un fournisseur, le moteur de "matching" reçoit une annonce du service en DAML-S. Le moteur de "matching" fait alors appel à un traducteur. Ce dernier construit une description du service en UDDI, l'enregistre et renvoie, au moteur de "matching", un identifiant. Le moteur de "matching" mémorise alors cet identifiant avec l'annonce du service en DAML-S.

Lors d'une requête de service, le moteur de "matching" peut sélectionner, à partir du profil hypothétique du service demandé, un service qui lui correspond au mieux. Le moteur de "matching" renvoie alors la description DAML-S du service correspondant ainsi que son identifiant dans le registre UDDI.

4 Interaction avec WSDL

DAML-S fournit une spécification de haut niveau, c'est-à-dire au niveau "application". Cette spécification est intéressante pour la découverte et la composition, mais elle est insuffisante pour l'invocation d'un service web. WSDL, quant à lui, fournit une spécification de plus bas niveau (niveau "interface") qui permet de réaliser l'invocation d'un service. Ces deux langages paraissent donc complémentaires. C'est pourquoi, certains articles [1,3] proposent de réaliser le "grounding" d'un service à l'aide de WSDL.

Effectuer un "grounding" en WSDL nécessite deux étapes. La première consiste à créer une sous-classe de la classe "ServiceGrounding" propre à l'usage de WSDL. La seconde étape nécessite un mécanisme de correspondance entre certains concepts de DAML-S et certains de WSDL. Il existe trois grandes règles de correspondance :

1. un processus atomique en DAML-S correspond ($\rightleftarrows$) à la notion d'opération de WSDL. Il existe quatre situations où cette correspondance peut être obtenue. Ces quatre situations sont résumées à la table 1.
2. DAML-S autorise d'avoir un ensemble d'entrées ou de sorties. Cette notion n'existe pas en WSDL, c'est pourquoi un ensemble d'entrées ou de sorties sera traduit, en WSDL, par un message unique.
3. la notion de type des entrées et sorties de DAML-S correspond à la notion de type abstrait de WSDL. Cette correspondance permet la traduction des entrées et des sorties.

Grâce à ces deux étapes, il est aisé de créer un "grounding" pour un service DAML-S en WSDL. Cependant, il existe des limitations. Il ne sera pas toujours possible de définir un "grounding" en WSDL. Ces limitations et d'autres restrictions plus générales de DAML-S seront exposées dans le point suivant.

DAML-S		WSDL
Processus atomique de type entrées/sorties	$\rightleftarrows$	opération "request-response"
Processus atomique de type entrées seules	$\rightleftarrows$	opération "one-way"
Processus atomique de type sorties seules	$\rightleftarrows$	opération "notification"
Processus atomique de type entrées/sorties avec les sorties reçues avant l'envoi des données	$\rightleftarrows$	opération "solicit-response"

Tab. 1. Règles de correspondance entre DAML-S et WSDL.

5 Critique de DAML-S

La critique que nous allons émettre, sert à évaluer DAML-S dans son cadre d'application, mais aussi à créer d'éventuelles opportunités de développements futurs. C'est pourquoi, nous nous attarderons sur les problèmes liés à DAML-S afin de pouvoir offrir des perspectives d'amélioration. Cette critique reposera sur cinq grands points et une évaluation finale sera proposée dans la section suivante.

Premièrement, il faut remarquer que le modèle conceptuel est imprécis et peut mener à des inconsistances. Ainsi, les entrées, sorties, pré-conditions, effets du profil ne doivent pas forcément correspondre avec celles du "ProcessModel". Il peut en résulter, alors, une inconsistance qui sera détectée en cours d'utilisation du service [3,9]. Certains liens entre les modèles ne sont pas clairs et la documentation n'arrive pas toujours à les élucider. De plus, à l'heure actuelle, certaines parties de l'ontologie DAML-S sont peu ou non définies (par exemple, l'ontologie de ressources, celle du contrôle de processus et celle du temps). Ces trois critiques sont reconnues par la coalition de DAML-S. Remarquons, pour terminer ce point, qu'il existe une multitude de modèles possibles pour un service web mais il faudra veiller à en choisir un qui soit consistant.

Deuxièmement, le "mapping" en WSDL limite l'expressivité de DAML-S. En effet, la définition d'un grounding a souvent de l'influence sur la façon de décrire le service [9]. De plus, certains aspects de DAML-S ne peuvent pas être traduits par un "grounding" en WSDL [1]. Ainsi, les outputs conditionnels de DAML-S ne peuvent pas être traduits en WSDL car celui-ci n'admet qu'un seul message de sortie pour un processus.

Troisièmement, DAML-S est difficile à apprendre et à utiliser [9]. Effectivement, on trouve peu d'outils pour aider à la construction de modèles. De même, les exemples sont peu nombreux et limités. L'utilisation de DAML-S suppose des connaissances préalables telles que DAML/WSDL/SOAP. De plus, la documentation existe, mais aide peu à comprendre les aspects intuitifs de DAML-S. Enfin, la syntaxe lourde du "grounding" rend sa réalisation difficile.

Quatrièmement, la sémantique offerte par DAML+OIL n'est pas suffisante pour restreindre les interprétations des modèles à celles voulues et uniquement à celles-là [1]. C'est pourquoi, certains chercheurs essaient de définir des sémantiques opérationnelles de DAML-S basées sur les réseaux de Petri, le "situation calculus" ou encore le "π-calculus" [8,7].

Cinquièmement, certains mécanismes de DAML-S posent des problèmes de réalisation pratique. Ainsi, le "matching" nécessite un processus très complexe [2]. Cela entraîne une perte de performance. Pour améliorer la performance, il faut alors diminuer la qualité du "matching". Cet aspect est un dilemme important étant donné que cette activité doit se faire aussi rapidement que possible et avec un degré de précision le plus élevé possible. La composition de service, quant à elle, nécessite aussi des mécanismes complexes. Certains auteurs essaient de rendre cette activité réalisable en transformant le profil en verbe, sujet, objet [6].

6 Evaluation et perspectives de DAML-S

En général, DAML-S est, sans aucun doute, très utile. En effet, il donne la possibilité de rendre automatiques des tâches de découverte, d'invocation, de composition et de "monitoring" de services web. Même si à l'heure actuelle, tous ces objectifs ne sont pas encore concrétisés, les développements futurs combleront très certainement ces lacunes.

Évaluons maintenant DAML-S en reprenant systématiquement ses quatre objectifs afin de voir lesquels sont partiellement ou complètement atteints. Le premier est de permettre la découverte automatique d'un "Web Service" sur Internet. On peut dire que cet objectif est partiellement réalisé. Effectivement, il existe des algorithmes de "matching" efficaces. Cependant, des recherches doivent encore être effectuées pour les rendre plus rapides ou pour en trouver de meilleurs qui possèdent déjà cette qualité. Le second but qui consiste à pouvoir invoquer automatiquement le service web, est dépendant du choix du "grounding". Dans le cas d'un "grounding" en WSDL, cet objectif est partiellement atteint. En effet, on peut définir, pour certains services, un "grounding" à l'aide de WSDL. Cela permet ensuite d'invoquer le service. Néanmoins, tous les services ne peuvent bénéficier d'un "grounding" en WSDL. Nous avons déjà exposé ce point dans les critiques. Des solutions doivent être trouvées en proposant, par exemple, des possibilités de "grounding" vers d'autres langages que WSDL. Cela constitue une perspective d'investigation. Le troisième objectif est la composition automatique de service. Celui-ci n'est pas encore réalisé car les algorithmes en cours de développement nécessitent des bases théoriques et sémantiques supplémentaires. Certaines recherches visent à combler ce manquement [8,7]. Enfin, le dernier objectif qui doit permettre de surveiller l'état d'avancement d'un service web, n'est pas atteint par manque de définition de DAML-S. En effet, l'ontologie, support de cette activité, n'est pas encore définie ("Process Control Ontology"). Cette lacune sera sans doute comblée dans les versions futures de DAML-S.

Du point de vue de la facilité de mise en oeuvre, DAML-S en fait défaut. En effet, il existe peu d'outils permettant de servir de support à la création de description de services web en DAML-S. De plus, la définition d'un "grounding" est une tâche très difficile et très complexe syntaxiquement. Ces éléments entraînent un manque d'utilisation de DAML-S en pratique [9]. Des outils de support sont nécessaires et ouvrent donc un pôle de développement.

7 Conclusion

DAML-S constitue une étape importante vers une utilisation automatique des ressources de l'Internet. Même si ce langage de description de services web n'est pas encore assez mature pour remplir tous ses objectifs, il constitue une bonne base, qui avec les développements nécessaires, permettra d'aboutir à ses objectifs. Il faudra encore des recherches additionnelles afin de rendre DAML-S utilisable. De plus, le développement d'outils facilitera la mise en oeuvre pratique

de description en DAML-S. En effet, la complexité de déploiement s'est traduite, dans la réalité, par une très faible utilisation de DAML-S [9].

OWL-S [10] est le successeur de DAML-S. Cette ontologie se veut plus complète que DAML-S et veut pallier certaines faiblesses de ce dernier. Si OWL-S est plus complet, certaines des lacunes de DAML-S ne sont toujours pas comblées. Ainsi, par exemple, le manque de consistance, de sémantique et d'outils font toujours défaut à OWL-S. Ce dernier ouvre, cependant, des perspectives de recherche et de développement pour tout chercheur désireux de voir, un jour, naître un "Web" riche en sémantique dont les services offerts seraient nombreux et faciles d'accès grâce à des agents de découverte, d'invocation, de composition et de surveillance des multiples services présents, à l'heure actuelle, sur le net.

Références

1. Ankolenkar, A., Burstein, M., Hobbs, J.R., Lassila, O., Martin, D.L., McDermott, D., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T.R., Sycara, K. : DAML-S : Web service description for the semantic web (2002)
2. Paolucci, M., Kawmura, T., Payne, T., Sycara, K. : Semantic matching of web services capabilities. In : First Int. Semantic Web Conf. (2002)
3. Ankolenkar, A., Burstein, M., Hobbs, J.R., Lassila, O., Martin, D.L., McDermott, D., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T.R., Sycara, K., Son, T.C., Zeng, H. : DAML-S : Semantic markup for web services (2003)
4. Andersen, B. : What is an ontology ? (2001)
5. Pilioura, T., Tsalgatidou, A., Batsakis, A., Hopkins, J. : Using wsdl/uddi and daml-s in web service discovery (2003)
6. Sheshagiri, M., desJardins, M., Finin, T. : A planner for composing services described in DAML-S (2003)
7. Ankolekar, A., Huch, F., Sycara, K.P. : Concurrent execution semantics for DAML-S with subtypes. In : Coordination Models and Languages. (2002) 14–21
8. Narayanan, S., McIlraith, S.A. : Simulation, verification and automated composition of web services (2002)
9. Sabou, M., Richards, D., van Splunter, S. : An experience report on using DAML-S (2003)
10. Ankolenkar, A., Hobbs, J.R., Lassila, O., Martin, D.L., McDermott, D., McIlraith, S.A., Paolucci, M., Payne, T.R., Sycara, K., Srinivasan, N., Solanki, M., McGuinness, D., Denker, G., Parsia, B., Sirin, E., Sabou, M. : OWL-S : Semantic markup for web services (2003)