Interopérabilité des systèmes multi-agents à l'aide des services Web

Amal El Fallah-Seghrouchni* — **Serge Haddad**** — **Tarak Melitti**** — **Alexandru Suna***

* *LIP6-Université Paris 6*
8, Rue du Capitaine Scott
75015 Paris
{Amal.Elfallah,Alexandru.Suna}@lip6.fr

** *LAMSADE-Université Paris Dauphine*
Place du Maréchal de Lattre de Tassigny
75016 Paris
{haddad,melitti}@lamsade.dauphine.fr

RÉSUMÉ. *Nous présentons un cadre conceptuel et architectural pour l'interopérabilité des systèmes multi-agents (SMA) hétérogènes. Notre approche est fondée sur les services Web, qui permettent à des applications d'exposer leurs fonctionnalités au travers d'interfaces standardisées. Ils favorisent ainsi une architecture orientée services, intégrant des systèmes hétérogènes complexes, fortement distribués et pouvant coopérer sans recourir à une intégration spécifique et coûteuse. Dans notre modèle, les agents publient leurs capacités en tant que services web qui peuvent être utilisés par d'autres agents et ce indépendamment des caractéristiques conceptuelles (architecture, modèle d'interaction, etc.) et techniques (plate-forme, langage de programmation, etc.) de ces agents. L'architecture proposée ainsi que les concepts introduits ont été testés et validés en utilisant le langage CLAIM et la plate-forme SyMPA.*

ABSTRACT. *In this paper we present a conceptual and architectural framework for the multi-agent systems' interoperability. Our approach uses the Web Services, that allow to applications to present their functionalities using standardized interfaces. They propose a service-oriented architecture, containing complex heterogenous distributed systems that can cooperate without a specific and costly integration. In our model, the agents publish their abilities as Web services that can be used by other agents, independently of conceptual (e.g. architecture) or technical (e.g. platform, programming language) aspects. The proposed architecture and concepts have been tested and validated using the CLAIM language and the SyMPA platform.*

MOTS-CLÉS : *systèmes multi-agent, services Web, interopérabilité*

KEYWORDS: *multi-agent systems, Web services, interoperability*

1. Introduction

Dans ce papier, nous présentons un cadre conceptuel et architectural permettant l'interopérabilité entre systèmes multi-agents (SMA) hétérogènes. L'interopérabilité a pour objectif de permettre à plusieurs systèmes logiciels, de même nature ou hétérogènes, de communiquer et de coopérer afin de minimiser les coûts d'intégration de ces systèmes. L'interopérabilité repose sur des normes techniques (à la base des standards) qui définissent des exigences accompagnées de recommandations permettant à deux systèmes qui satisfont aux exigences de dialoguer et d'évoluer librement tant qu'ils respectent la norme définissant leurs interfaces.

Longtemps étudiée, l'interopérabilité est de plus en plus requise du fait du déploiement d'un grand nombre d'applications sur Internet et de leur potentiel d'interaction et de coopération. Elle peut être abordée de façon statique (compilation de deux applications à travers un bus CORBA par exemple) ou de manière dynamique (invocation de service par exemple). Trois niveaux d'interopérabilité peuvent être distingués : le niveau technologique, le niveau syntaxique et le niveau sémantique. En tant que systèmes logiciels, les systèmes multi-agents sont également confrontés au problème de l'interopérabilité d'autant plus que leur vocation à interagir et à coopérer de façon autonome est l'essence même de leur existence. Les agents cognitifs exhibent souvent un comportement orienté but. De ce fait, l'interopérabilité ne peut se limiter à une simple invocation de services mais nécessite la mise en place : 1) de mécanismes sophistiqués de recherche de services capables de satisfaire des besoins ; 2) et de modèles d'interaction complexes.

L'interopérabilité des SMA a d'ailleurs été promue par différentes institutions (ou consortiums) dont la FIPA[1] qui déploie d'énormes efforts afin de proposer des normes (spécifications) et des standards qui assurent une interopérabilité à différents niveaux des architectures SMA[FLO 99, LYE 02]. On peut citer à titre d'exemple : FIPA-ACL [LAB 99] au niveau communication, DAML+OIL [HOR 01] au niveau sémantique, AAA[2] au niveau architectural, etc. La FIPA a également encouragé le développement de différents plates-formes respectant les spécifications qu'elle a proposées (JADE [NWA 99] et ZEUS [BEL 99] par exemple[3]).

Mais les limites des standards de la FIPA résident dans le fait que l'interopérabilité est conditionnée par la compatibilité des SMA avec les spécifications FIPA : au lieu d'assurer l'interopérabilité entre SMA hétérogènes, la FIPA tend à rendre tous les SMA compatibles (via ses spécifications). De plus, le niveau de cognition exigé par les spécifications FIPA (par exemple les actes du langage dans le cas de FIPA-ACL) rend difficile une interopérabilité entre SMA hétérogènes.

---

1. http ://www.fipa.org
2. Abstract Agent Architecture
3. plates-formes compatibles FIPA : http ://www.fipa.org/resources/livesystems.html

Afin de maximiser l'intégration de systèmes hétérogènes et de favoriser une interopérabilité au moindre coût, il est nécessaire de minimiser les exigences en termes de spécifications.

Dans le monde du génie logiciel, et comme en témoignent les efforts de normalisation et de développements autour des services Web [CUR 01], ces derniers se présentent aujourd'hui comme un support crédible permettant à des applications d'exposer leurs fonctionnalités au travers d'interfaces standardisées et de plus en plus éprouvées. Les services Web ont pour vocation de favoriser une architecture orientée services, intégrant des systèmes hétérogènes complexes, fortement distribués et pouvant coopérer sans recourir à une intégration spécifique et coûteuse (figure 1). La souplesse introduite par les services Web permet d'envisager de nouvelles formes de collaboration entre applications distantes. Néanmoins, les services Web ne représentent qu'un point de départ vers l'interopérabilité. Ils soulèvent des problèmes qui sont encore du domaine de la recherche.

Aujourd'hui on assiste à un rapprochement entre les SMA et les services Web. Ce rapprochement se manifeste selon différentes directions dont les principales sont les suivantes.

– L'utilisation des SMA en tant qu'entité médiatrice dans le modèle fonctionnel des services Web (voir la section 2). La médiation intervient à plusieurs niveaux. Par exemple, les auteurs de [MOS 03] proposent des SMA «proxy» qui facilitent la localisation de services Web tandis que dans [RIC 03, SHE 03, LAU 03] des SMA de planification et de composition de services Web sont présentés.

– L'utilisation des services Web comme cadre architectural et technologique pour mettre en place des SMA accessibles à travers le Web. Dans ce type d'application on retrouve des applications agents qui offrent leurs capacités à travers des services Web. Ici on distingue deux catégories qui diffèrent essentiellement par leur conception :

- Une conception intégrée : ce sont des services Web développés suivant un modèle agent afin de réaliser des tâches complexes telles que la gestion des transactions ou des interactions commerciales (voir par exemple [JIN 03]). Ainsi dans [PET 03], on propose un ensemble de primitives de communication empruntées des langages de communication agents supportant des interactions dynamiques.

- Une conception découplée : à partir d'un SMA donné *a priori*, une couche à base de services Web rend les capacités de l'agent accessibles à travers le Web que ce soit à d'autres agents de SMA ou à des applications clientes traditionnelles. Par exemple [LYE 03] présente des outils qui traduisent des descriptions de service (SD) d'agent FIPA en services Web WSDL.

Le travail présenté ici participe à l'approche de conception découplée. A la différence de [LYE 03] qui se borne à une traduction syntaxique d'un langage d'interface vers un autre (SD vers WSDL) notre démarche inclut les aspects originaux suivants :

– Les algorithmes de synthèse de comportement d'un service et d'un client supportant les fonctionnalités rencontrées dans les protocoles d'interaction (sélection des

agents, suivi d'exécution, respect des contraintes temporelles, etc.). L'intérêt majeur de ces algorithmes est de générer des modèles de comportement à partir d'éléments purement déclaratifs.

– Notre environnement s'accompagne d'une méthode qui permet au concepteur du SMA d'identifier les parties à réécrire et de choisir le niveau d'intégration à partir d'un compromis entre le coût de la réécriture et l'interopérabilité du SMA ainsi modifié.

Basé sur les services Web, l'environnement d'interopérabilité que nous proposons, appelé Web-MASI (Web-MAS Interoperability) vise à minimiser les coûts d'intégration tout en minimisant les exigences en termes de spécification des SMA. L'interopérabilité repose sur deux éléments : une architecture qui encapsule les SMA dans le modèle de fonctionnement des services Web et un module d'interopérabilité qui constitue l'interface entre les SMA et l'environnement des services Web. L'idée qui sous-tend ce travail est que les agents (comme d'autres applications coopératives via le Web) peuvent publier leurs capacités en tant que services Web invocables par d'autres agents et ce indépendamment des caractéristiques conceptuelles (architecture, modèle d'interaction, etc.) et techniques (plate-forme, langage de programmation, etc.) de ces agents. Les éléments d'interopérabilité que nous avons identifiés sont les buts traduisant des besoins, les capacités représentant des services et les protocoles de coopération pouvant rendre effective l'interaction entre agents demandeurs et agents fournisseurs de services. L'architecture proposée ainsi que les concepts introduits ont été testés et validés en utilisant le langage CLAIM [FAL 03] et la plate-forme SyMPA [SUN 04].

2. Rappel des principes des services Web

2.1. *Modèle de fonctionnement*

Le modèle des services Web repose sur une architecture orientée service [BOO 03]. Celle-ci fait intervenir trois catégories d'acteurs : les fournisseurs de services (i.e. les entités responsables du service Web), les clients qui servent d'intermédiaires aux utilisateurs de services et les annuaires qui offrent aux fournisseurs la capacité de publier leurs services et aux clients le moyen de localiser leurs besoins en terme de services. La dynamique de l'architecture se décompose ainsi : la publication du service par le fournisseur dans un annuaire, la localisation du service par le client et enfin, l'interaction entre le client et une instance d'exécution du service. Cette dynamique est normalisée à travers un certain nombre de standards : un protocole abstrait de description et de structuration des messages, SOAP[4] [SOA 00], une spécification XML qui permet la publication et localisation des services dans les annuaires, UDDI[5] [UDD 02] et un format de description des services Web publiées dans les annuaires, WSDL[6] [WSD 01]. Un service WSDL est composé d'un ensemble d'opérations élémentaires, chacune dé-

4. Simple Object Access Protocol
5. "Universal Description, Discovery and Integration
6. Web Services Description Language

crite par un flux de messages échangés entre le client et le service. De plus, WSDL spécifie les protocoles de transport et d'échange des messages.

La figure 1 synthétise les aspects statiques et dynamiques du modèle des services Web.

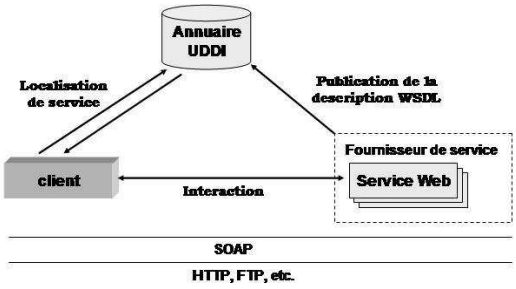

Figure 1. *Modèle de fonctionnement de l'architecture Services Web*

Les services Web constituent un cadre robuste pour assurer l'interopérabilité entre des applications hétérogènes, accessibles en ligne, car ils n'imposent pas de restriction sur les caractéristiques techniques de l'application qui implémente le service ; ils proposent une représentation homogène du comportement observable du service (i.e. du point de vue du client).

2.2. *Services Web complexes : aspects opérationnels et sémantiques*

Cependant, l'état actuel du modèle présente des limites conceptuelles dont le dépassement constitue deux axes de recherche complémentaires. D'une part, les taxonomies des annuaires ne sont pas assez expressives pour permettre une correspondance fine entre les besoins des utilisateurs et les services proposés. Les recherches actuelles s'inscrivent dans le cadre du Web sémantique et sont fortement liées à la spécification d'ontologies appropriées (DAML-S), [SIV 03]. D'autre part, la sémantique associée à WSDL n'est pas assez expressive pour supporter certains types d'applications qui nécessitent une interaction longue contrôlée par un modèle explicite de processus. Plusieurs propositions d'extension de WSDL ont été proposées pour supporter les services Web complexes, tels que Xlang, WSFL, etc [STA 03][BEN 01]. Ces langages proposent un jeu de balises définissant des opérateurs qui s'appliquent de manière modulaire sur les unités de base d'échange de messages (les opérations WSDL). La particularité de Xlang se manifeste à travers deux aspects. Premièrement, il offre un ensemble d'opérateurs temporels tels que les opérateurs de dépassement de délai (i.e. « timeout ») et les contraintes d'échéances. Deuxièmement, le comportement du service est décrit en fonction de ses aspects observables par le client. En voici les principaux

constructeurs[7]. On désigne par $!o[m]$ et $?o[m]$ les opérations WSDL respectivement d'envoi et de réception de message et par $r[e]$ la levée d'une exception e.

– $P; Q$ le constructeur de séquence, où le service se comporte comme P ; une fois P terminé, il se comporte comme Q.

– $switch[\{P_i\}_{i \in I}]$ suite à un processus de choix interne, le service se comporte de manière indéterministe, vu du client, suivant une des descriptions P_i.

– Le constructeur $while[P]$ représente un service qui se comporte itérativement comme le service P jusqu'à ce qu'une condition interne au service (opaque au client) ne soit plus vérifiée.

– Le constructeur $all[\{P_i\}_{i \in I}]$ définit un service qui exécute parallèlement les services P_i. Il se termine quand tous les processus P_i se terminent.

– $pick[\{(m_i, P_i)\}_{i \in I}, d, Q]$ suite à la réception d'un message m_i, se comporte comme le service P_i. Si aucun des messages m_i n'est parvenu avant d unités de temps, le service se comporte comme le service Q.

– $context[P, E]$ le constructeur $context$ définit un service P gardé par un certain nombre d'événements définis dans le bloc d'exception E.

La description observable du service entraîne (par construction) un non déterminisme dans son comportement, ce qui rend le processus d'interaction non trivial pour les clients traditionnels. Dans un précédent travail [HAD 04] nous avons résolu ce problème. Tout d'abord nous avons défini une sémantique formelle de Xlang au moyen des algèbres de processus temporisés afin de représenter le comportement du service par un automate temporisé. Puis nous avons défini une relation d'interaction que doivent vérifier le client et le service Web. Enfin nous avons conçu un algorithme qui, prenant en entrée l'automate temporisé du service Web, soit construit l'automate d'un client correct soit détecte l'ambiguïté du service c'est à dire l'impossibilité d'un tel client. Ce travail a donné lieu à une implémentation d'un client générique capable d'interagir avec les services Web complexes. Le module de client générique est utilisé dans la construction du module d'intégration comme nous le verrons dans la suite.

3. Environnement d'intégration des SMA hétérogènes

3.1. *L'architecture d'interopérabilité*

L'environnement d'interopérabilité que nous proposons repose sur deux élements clés : une architecture qui encapsule les SMA dans le modèle de fonctionnement des services Web et un module d'interopérabilité qui constitue l'interface entre les SMAs et l'environnement services Web. Dans notre architecture d'intégration, les agents représentent à la fois les fournisseurs et les clients du service. Ils utilisent les annuaires UDDI afin de publier et localiser leurs capacités en vue d'être découvertes et utilisées par d'autres agents et ceci de manière modulaire et uniforme. Le Module d'Intégration

7. présentés dans une syntaxe simplifiée (sans la lourdeur de la syntaxe XML)

(MI) offre aux différents systèmes multi-agents une bibliothèque afin de synthétiser, publier, localiser et invoquer des services Web. L'invocation et le suivi d'exécution du service sont réalisés par notre client générique contenu dans le MI. Tels que nous les avons conçus, les services Web générés à partir des capacités des agents sont invocables à la fois par d'autres agents ou par des utilisateurs disposant d'un client générique et vice versa.

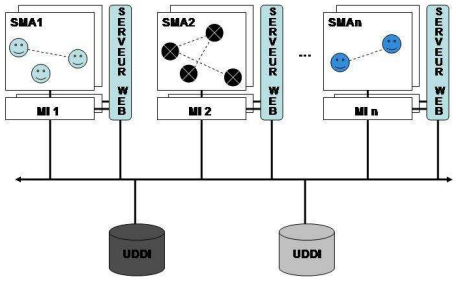

Figure 2. *Architecture générale de l'environnement*

3.2. *Exigences / Pre-réquis*

Nous avons porté notre effort sur la minimisation des exigences fonctionnelles des SMA et ceci afin de minimiser le coût d'intégration et par conséquence de couvrir le plus largement possible les différents modèles de SMA. Toutefois, ces SMA doivent présenter certaines caractéristiques que nous décrivons ci-dessous. Celles-ci se définissent soit au niveau du SMA, soit au niveau de l'agent.

Exigence au niveau des SMA
Chaque agent doit disposer d'une identité unique au sein du SMA. Cette caractéristique est essentielle pour le bon fonctionnement du MI. En effet, ce module doit identifier le propriétaire d'une capacité publiée pour jouer son rôle d'intermédiaire entre les clients et les prestataires. Dans la plupart des systèmes multi-agents, un tel mécanisme de désignation est présent (cas du modèle de communication adressé). Dans les systèmes mono-agent cette caractéristique est implicitement garantie.

Exigence au niveau des agents
Les pré-requis d'un agent portent globalement sur les conditions dans lesquelles l'agent est amené à interagir et sur son processus d'interaction. Nous décrivons les éléments qui doivent intervenir dans le fonctionnement de l'agent :
1. **Les buts**
2. **Les capacités** Notre architecture requiert qu'un agent soit une entité autonome capable d'effectuer un certain nombre de tâches (capacités) chacune munie d'une description couvrant les aspects liés à l'invocation et à l'exécution (voir 3.3.1).

3. **L'offre et la demande** Les agents doivent prendre l'initiative de satisfaire leurs buts de façon externe au SMA et de publier les capacités qu'ils veulent partager.

Les exigences fonctionnelles sont définies à un niveau suffisamment abstrait afin de permettre à chaque SMA de les mettre en place en prenant en compte leurs caractéristiques conceptuelles et techniques. Par exemple, chaque agent est libre de la stratégie de publication de capacités ou d'utilisation d'un service Web.

3.3. *Dynamique de l'environnement d'intégration*

Un SMA distribué est composé d'agents s'exécutant sur plusieurs ordinateurs connectés via un réseau. Le module d'interopérabilité (MI) que nous proposons est déployé sur chacun des ordinateurs susceptible d'accueillir des agents. Ce module est composé de deux sous-modules qui accomplissent respectivement les fonctions de publication (P) d'un service Web à partir d'une capacité d'un agent et celles de recherche et d'invocation (RI). La figure 3 présente le fonctionnement du module d'interopérabilité et les interactions entre les agents, ce module, les serveurs Web (SW) et les annuaires UDDI.

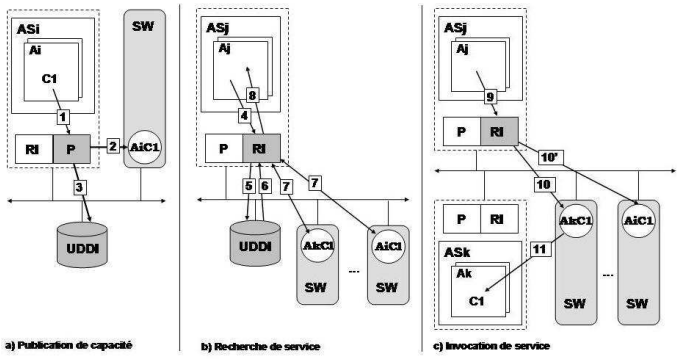

Figure 3. *Fonctionnement du module d'interopérabilité (MI)*

a) Publication
Ce sous-module implémente l'algorithme de synthèse de service et fait appel à un client UDDI pour la publication. Voici son fonctionnement :
1. Un agent A_i décide de publier, en utilisant le module de publication (P) une ou plusieurs de ses capacités (C_1 sur la figure).
2. À partir de la description de la capacité, le module P crée un service Web (A_iC_1) et le déploie sur le serveur Web (SW) local (voir 3.3.1).
3. La description du service (qui inclut l'effet de la capacité correspondante) est publié dans un registre UDDI.

b) Recherche et invocation
Ce sous-module utilise le module de synthèse de client et offre les services nécessaires à la localisation et l'invocation de service. Voici son fonctionnement :
4. Lorsqu'un agent (A_j) L'agent utilise les fonctionnalités du module de recherche de RI afin de localiser des services Web qui pourraient satisfaire son besoin. L'agent doit spécifier s'il désire choisir parmi plusieurs services selon un ensemble de critères ou s'il laisse le module RI choisir.
5. Le module RI interroge le registre UDDI.
6. Le module RI obtient de la part du registre une liste de descriptions de services Web correspondant à ses critères de recherche.
7. Le module RI commence l'invocation des services découverts afin d'obtenir leurs divers attributs et ceci de manière autonome (i.e. sans intervention de l'agent).
8. Si l'agent a exprimé son désir de sélectionner les services appropriés, le module de recherche envoie à l'agent ces descriptions de services. Sinon, le module RI choisit un service (de manière aléatoire) et continue avec l'étape 10.
9. L'agent (A_j) précise au module RI le choix d'un (ou plusieurs) service Web, choisi parmi ceux dont la description lui a été envoyée par le module RI dans l'étape 8. Sans que cela soit nécessaire, lors de l'intégration d'un SMA un module de sélection de services peut être ajouté à chaque agent.
10. Le module RI envoie simultanément des messages de confirmation aux services selectionnés et des messages d'annulation aux autres services considérés (sur la figure, l'agent A_j a choisi le service A_kC_l correspondant à la capacité C_l de l'agent A_k).
11. De son côté, le service transmet la traduction de ce message dans le format de l'agent fournisseur de la capacité, pour démarrer son exécution.
Après l'exécution de la capacité, l'agent demandeur est informé du résultat par l'intermédiaire du module d'intégration.

Un des points forts de notre architecture est que le service Web déployé ne se limite pas à l'invocation de la capacité mais intègre la phase de négociation (avec ou sans intervention de l'agent demandeur) et du suivi d'exécution. Ceci est rendu possible par l'algorithme de synthèse de service à partir d'une déclaration de capacité.

3.3.1. *Présentation informelle de l'algorithme de synthèse de service*

L'algorithme de synthèse prend en entrée une description XML de la capacité et de ses attributs. Le choix de la grammaire associée est mineur. Aussi nous nous concentrons sur les éléments de cette description qui se regroupent en trois catégories.

– **Description de l'agent :** l'identité de l'agent fournisseur de la capacité ainsi que des informations optionnelles (e.g. la mobilité, etc.) neutres du point de vue de l'algorithme et dont l'interprétation se fera au niveau des agents clients.

– **Description de la capacité :** une description sémantique composée de quatre éléments XML, son nom, ses pré et post-conditions, et les messages nécessaires au processus d'interaction. Les pré et post conditions font référence à un ou plusieurs espaces de noms destinés à l'agent client. Si une pré-condition est présente, l'algorithme de synthèse prend en compte la possibilité d'un refus d'exécution. La post-condition

est l'élément clef de la recherche dans les registres UDDI. L'attribut message fait référence à la spécification WSDL et Xlang. Actuellement notre environnement supporte les capacités nécessitant un message d'entrée (i.e. une requête) et un éventuel message de sortie (i.e. une réponse).

– **Description de l'interaction :** cette partie est facultative mais améliore les conditions de l'interaction. Elle est composée d'attributs statiques et dynamiques. Les attributs statiques définissent les aspects de l'exécution de la capacité connus *a priori* tels que les contraintes temporelles, la possibilité d'annulation en cours d'exécution, etc. Les valeurs des attributs dynamiques sont déterminées au moment de l'interaction (e.g. la disponibilité de l'agent, la qualité de service de la tâche, etc). La détermination des valeurs de ces attributs requiert un dialogue entre le service Web et l'agent fournisseur de la capacité (ou éventuellement avec l'environnement). Ainsi chacun de ces attributs doit inclure la manière d'obtenir sa valeur. La version actuelle de notre prototype associe à chaque attribut dynamique le message nécessaire à sa récupération.

La figure 4 décrit sur la gauche un extrait de la description d'une capacité et sur la droite un extrait du service Web généré.

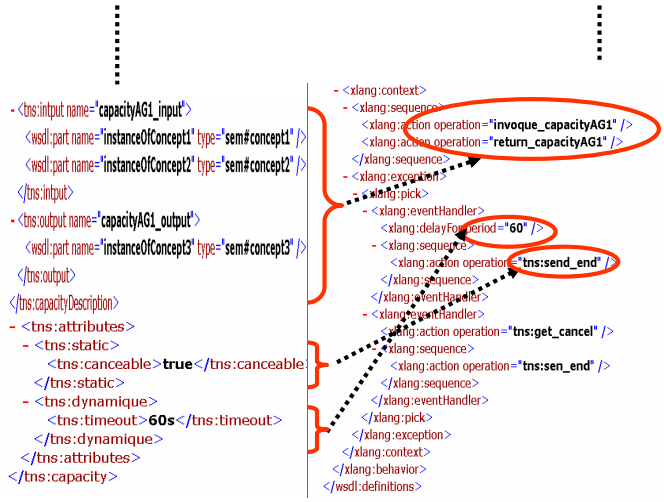

Figure 4. *Exemple de résultat de l'algorithme de synthèse*

Le service Web généré est un service Web décrit en Xlang. L'invocation d'une capacité comporte deux phases : une phase de négociation au cours de laquelle les agents entament une interaction afin de récupérer les informations sur les attributs statiques et dynamique et une deuxième phase liée à l'exécution de la capacité.

1) **Phase de négociation :** Elle débute par une réception d'un message de demande d'interaction. Une fois ce message reçu, le service récupère les valeurs des attributs dynamiques. Selon ces valeurs, deux cas sont alors possibles : soit l'interaction est possible et le service envoie un message contenant les caractéristiques de la capacité

soit le service envoie un message informant que l'interaction est actuellement impossible. Voici un exemple relativement générique de cette phase (en utilisant la syntaxe XLANG introduite précédemment).
$negociation$:?$o[demande\_interaction]; switch($!$o[attributs]; execution,$
$!o[rejet])$

2) **Phase d'exécution :** La structure même du service XLANG (appelé $execution$ dans l'exemple précédent) dépend de la description de la capacité. L'interaction nécessaire à l'exécution de la capacité est encapsulée dans un bloc $context$ gardé par des événements générés à partir des attributs tels qu'un message d'annulation pour les capacités qu'on peut avorter en cours d'exécution, un délai qui représente le temps consacré à l'exécution de la capacité, etc. Le corps du $context$ est constitué par les flux de messages de l'exécution de la capacité. En voici un exemple. :
$execution : context[?o[invocation]; switch(!o[resultat]; !o[probleme]),$
$[pick[(annulation, !o[avorte]), (delai\_max, !o[timeout])]$

Pour chaque description de capacité un service Web est déployé et publié. L'algorithme de synthèse génère le service Web correspondant aux deux phases et le fichier WSDL contenant tous les messages nécessaires et les opérations WSDL. Le fichier Xlang est ensuite généré de manière dynamique comme décrit précédemment. Les fichiers WSDL et Xlang ainsi que le fichier de description de la capacité sont publiés sur un registre UDDI.

4. Expérimentation sur CLAIM et SyMPA

Nous présentons maintenant le processus d'intégration de la plate-forme SyMPA (qui supporte des agents CLAIM) afin de valider notre approche.

CLAIM [FAL 03] est un langage déclaratif de programmation orientée agent qui combine des éléments cognitifs, spécifiques aux agents intelligents, des primitives de communication et des primitives de mobilité inspirées du calcul des ambients [CAR 98]. CLAIM permet la conception et l'implantation de SMA distribués, situés sur des ordinateurs (sites) connectés via un réseau. Un agent CLAIM a des composantes cognitives, comme les connaissances (sur les autres agents et sur le monde), les buts et les capacités qui permettent un comportement réactif ou un comportement orienté but.

Les agents définis en CLAIM satisfont les exigences précédemment définies (voir 3.2). Ils sont guidés par des buts, qui peuvent être atteints au moyen des capacités. L'invocation d'une capacité est déclenchée par la réception d'un message si une condition (optionnelle) est vérifiée. L'exécution d'une capacité engendre des effets, sorte de post-condition. Les connaissances, les buts, les messages, les conditions et les effets des capacités sont représentés comme des propositions ayant un nom (prédicat) et une liste de paramètres ou d'arguments. Le langage est supporté par une plate-forme

d'agents mobiles, appelée SyMPA[8] [SUN 04]. Le processus d'intégration consiste à développer une couche supplémentaire à SyMPA pour l'interfacer avec le module d'interopérabilité

Publication des capacités d'un agent CLAIM en tant que service Web

Nous avons d'abord développé une méthode qui à partir d'une capacité CLAIM génère sa description au format Web-MASI. Ainsi, le message, la condition et l'effet de la capacité sont pris en compte. L'intégration de la phase de publication s'est limitée ensuite à l'ajout d'un appel de cette méthode par l'agent. Les capacités CLAIM n'ont pas des attributs dynamiques et sont atomiques (ne peuvent pas être interrompues). Par contre, la pré-condition d'une capacité est prise en compte d'une part par l'attribut éponyme qui indique au client la possibilité d'un échec et d'autre part dans la description UDDI qui contient les propositions représentant l'effet de la capacité et la pré-condition.

Recherche d'un service selon un critère

Sans le module d'interopérabilité, CLAIM offrait aux agents des mécanismes pour présenter leurs capacités ou pour chercher des capacités provenant d'autres agents CLAIM. Néanmoins, ces mécanismes ne sont pas implicites et chaque programmeur peut les intégrer ou non à ses agents.

Grâce au module d'interopérabilité, un agent peut publier ses capacités et il peut demander des services à d'autres agents (provenant d'autres plates-formes).

Quand un agent essaie de satisfaire ses buts, un module (originel) de SyMPA va d'abord vérifier s'il a des capacités qui lui permettent de l'atteindre. Dans le cas contraire, il poursuit sa recherche auprès d'autres agents CLAIM. Nous avons ajouté un troisième type de recherche (par le biais du module d'interopérabilité) d'un service Web dont la description permet de déduire qu'il achèvera l'effet escompté. De plus, nous avons doté un agent CLAIM d'un module de sélection d'un service selon des critères propres à l'agent.

Invocation d'un service Web

Quand un agent CLAIM a choisi un service il fait appel au module d'interopérabilité pour l'invoquer. Nous avons aussi doté les agents CLAIM de la possibilité d'invoquer directement un service Web dont ils connaissent la description WSDL.

Exécution d'un service correspondant à une capacité d'un agent

Nous n'avons rien modifié à ce niveau en SyMPA. Ce sont des éléments créés pendant l'étape de publication qui ont une influence directe sur l'exécution d'une capacité publiée. Quand un service déployé sur un serveur Web (dans l'étape de publication) reçoit un message SOAP d'invocation, il doit transmettre cette invocation, dans un format CLAIM, à l'agent concerné. Si la condition de la capacité est vérifiée, l'agent exécute les processus correspondants. Après l'exécution, le résultat (y compris l'effet réalisé) est envoyé par le service Web au module d'invocation qui l'a demandé. Ce module envoie à son agent un message CLAIM pour l'informer de la réalisation de l'effet.

---

8. Système multi-plateforme d'agents : compatible avec les spécifications du standard MASIF de l'OMG.

En conclusion, l'utilisation du module d'interopérabilité pour CLAIM et SYMPA s'est avérée relativement facile. Cette utilisation a néanmoins impliqué le développement de petits modules de conversion entre le format Web-MASI et le format CLAIM et l'insertion dans le code des agents d'appels aux API fournies.

5. Conclusion et perspectives

Dans ce papier nous avons présenté un cadre conceptuel et architectural, fondée sur les services Web, pour l'interopérabilité des SMA hétérogènes. L'environnement d'interopérabilité proposé, Web-MASI, est composé de deux éléments : une architecture qui encapsule les SMA dans le modèle de fonctionnement des services Web et un module d'interopérabilité qui constitue l'interface entre les SMAs et l'environnement services Web. Ce travail ouvre des nombreuses perspectives. Dans un premier temps nous allons tester notre environnement d'interopérabilité sur d'autres SMA. Ensuite nous allons étendre l'algorithme de synthèse ainsi que la grammaire de description afin de représenter des capacités complexes. Enfin, nous envisageons d'intégrer des ontologies DAML-S dans la description sémantique des buts des agents.

6. Bibliographie

[BEL 99] BELLIFEMINE F., POGGI A., RIMASSA G., « JADE :A FIPA-compliant agent framework », *in Proceedings of PAAM*, London, 1999, p. 97-108.

[BEN 01] BENATALLAH B., DUMAS M., FAUVET M., RABHI F., « Towards Patterns of Web Services Composition », rapport, November 2001, Technical Report UNSW-CSE-TR-0111, The University of New South Wales Sydney, Australia.

[BOO 03] BOOTH D., HAAS H., MCCABE F., NEWCOMER E., « Web Services Architecture », aout 2003, http ://www.w3.org/TR/2003/WD-ws-arch-20030808/.

[CAR 98] CARDELLI L., GORDON A., « Mobile Ambients », *Foundations of Software Science and Computational Structures, LNAI*, vol. 1378, 1998, p. 140-155.

[CUR 01] CURBERA F., NAGY W. A., WEERAWARANA S., « Web Services : Why and How ? », *OOPSLA 2001 Workshop on Object-Oriented Web Services*, , 2001.

[FAL 03] FALLAH-SEGHROUCHNI A. E., SUNA A., « An Unified Framework for Programming Autonomous, Intelligent and Mobile Agents », *LNAI*, vol. 2691, 2003, p. 353-362.

[FLO 99] FLORES-MENDEZ B. R. A., « Standardization of Multi-Agent System Frameworks », *ACM Crossroads, 5(4)*, , 1999, ACM Press.

[HAD 04] HADDAD S., MELLITI T., MOREAUX P., RAMPACEK S., « Modelling Web Services Interoperability », *Proceedings of the Sixth International Conference on Entreprise Information Systems*, Porto, Portugal, 2004, p. 287–295.

[HOR 01] HORROCKS I., VAN HARMELEN F., PATEL-SCHNEIDER P., BERNERS-LEE T., BRICKLEY D., CONNOLLY D., M. DEAN S. D., FENSEL D., HAYES P., HEFLIN J., HENDLER J., LASSILA O., MCGUINNESS D., STEIN L., « DAML+OIL », Mars 2001, http ://www.daml.org/2001/03/daml+oil-index.

[JIN 03] JIN T., GOSCHNICK S., « Utilizing Web Services in an Agent Based Transaction Model (ABT) », *Workshop on Web services And Agent-based ingineering*, Melbourne, Australia, 2003.

[LAB 99] LABROU Y., FININ T., PENG Y., « Agent Communication Languages : The Current Landscape », *IEEE Intelligent Systems*, vol. 14, n° 2, 1999, p. 45–52.

[LAU 03] LAUKKANEN M., HELIN H., « Composing Workflows of Semantic Web Services », *Workshop on Web services And Agent-based ingineering*, Melbourne, Australia, 2003.

[LYE 02] LYELL M., « Interoperability, standards, and software agent systems », *23rd Army Science Conference*, 2002.

[LYE 03] LYELL M., ROSEN L., CASAGNI-SIMKINS M., NORRIS D., « On Software Agents and Web Services : Usage and Design Concepts and Issues », *Workshop on Web services And Agent-based ingineering*, Melbourne, Australia, 2003.

[MOS 03] MOSTEFAOUI S. K., MOSTEFAOUI G. K., « Towards A Contextualisation of Service Discovery and Composition for Pervasive Environments », *Workshop on Web services And Agent-based ingineering*, Melbourne, Australia, 2003.

[NWA 99] NWANA H. S., NDUMU D. T., LEE L. C., COLLIS J. C., « ZEUS : a toolkit and approach for building distributed multi-agent systems », *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, Seattle, WA, USA, 1999, ACM Press, p. 360–361.

[PET 03] PETRONE G., « Managing flexible interaction with Web Services », *Workshop on Web services And Agent-based ingineering*, Melbourne, Australia, 2003.

[RIC 03] RICHARDS D., VAN SPLUNTER S., BRAZIER F. M., SABOU M., « Composing Web Services using an Agent Factory », *Workshop on Web services And Agent-based ingineering*, Melbourne Australia, 2003.

[SHE 03] SHESHAGIRI M., DESJARDINS M., FININ T., « A Planner for Composing Services Described in DAML-S », *Workshop on Web services And Agent-based ingineering*, Melbourne, Australia, 2003.

[SIV 03] SIVASHANMUGAM K., VERMA K., SHETH A. P., MILLER J. A., « Adding Semantics to Web Services Standards », *Proceedings of the International Conference on Web Services ICWS '03, June 23 26, 2003, Las Vegas, Nevada, USA*, CSREA Press, 2003, p. 395-401.

[SOA 00] SOAP, « Simple Object Access Protocol (SOAP) 1.1 », rapport, may 2000, World Wide Web Consortium, http ://www.w3.org/TR/SOAP/.

[STA 03] STAAB S., VAN DER AALST W., BENJAMINS V. R., SHETH A., MILLER J. A., BUSSLER C., MAEDCHE A., FENSEL D., GANNON D., « Web Services : Been There, Done That ? », *IEEE Intelligent Systems : volume 18, ,* 2003, p. 72-85.

[SUN 04] SUNA A., FALLAH-SEGHROUCHNI A. E., « A mobile agents platform ; architecture, mobility and security elements », *ProMAS'04, workshop d'AAMAS, ,* 2004.

[UDD 02] UDDI, « Universal Description, Discovery and Integration », rapport, mar 2002, OASIS UDDI Specification Technical Committee, http ://www.oasis-open.org/cover/uddi.html.

[WSD 01] WSDL, « Web Services Description Language (WSDL) 1.1 », rapport, mar 2001, World Wide Web Consortium, http ://www.w3.org/TR/wsdl.