Une architecture pour la découverte et l'orchestration de services Web sémantiques

Une utilisation des ontologies en milieu industriel

Pierre Châtel

Thales Communications France, Laboratoire d'Informatique de Paris VI (LIP6)
1 à 5 avenue Carnot
91883 Massy Cedex - France
pierre.chatel@fr.thalesgroup.com

RÉSUMÉ. *Ce papier présente le travail effectué à Thales Communications France concernant la spécification et l'implémentation d'une architecture pour la découverte et l'orchestration de services Web sémantiques en fonction de leurs contraintes fonctionnelles. L'objectif principal étant d'améliorer l'interopérabilité des systèmes SOA hétérogènes et dynamiques, la particularité de cette architecture est de placer les ontologies au cœur de ce processus. Une implémentation de référence a été réalisée qui pourra par la suite être étendue pour s'accommoder des évolutions des besoins utilisateurs et du domaine. De plus, des travaux de recherche sont actuellement entrepris pour lui adjoindre le support des contraintes non fonctionnelles (ou extra-fonctionnelles) de qualité de service (QoS).*

ABSTRACT. *This paper presents the work that has been done at Thales Communications France regarding the specification and implementation of a semantic Web service discovery and orchestration platform based on services' functional constraints. The main objective was to increase interoperability in heterogeneous and dynamic SOA systems by putting ontologies at the center of this process. A reference implementation is provided that could easily be extended to accommodate an evolution of the domain or of users requirements. Also, ongoing research at Thales is focused on including non-functional (or extra-functional) constraint handling, like Quality of Service (QoS), to the framework.*

MOTS-CLÉS : *composition et orchestration de services, interopérabilité, processus métier, spécification statique, configuration dynamique, systèmes adaptatifs, services Web, ontologies, UDDI, BPEL, SAWSDL, OWL*

KEYWORDS: *web service composition and orchestration, interoperability, business processes, static specification, dynamic configuration, adaptive systems, web-services, ontologies, UDDI, BPEL, SAWSDL, OWL*

1. Introduction

De par son métier d'intégrateur de systèmes civils et militaires, Thales est amené à concevoir et manipuler des systèmes d'information à forte hétérogénéité : se pose alors le problème du maintien de l'interopérabilité entre les composants de ces systèmes.

Cette problématique est traditionnellement traitée au niveau syntaxique : en ce sens, l'interopérabilité reste superficielle et ne peut être maintenue qu'en imposant de fortes contraintes aux utilisateurs du système. Cela reste particulièrement difficile à maintenir, notamment lors du passage à l'échelle. De plus, la multitude de solutions ad hoc déjà existantes nous amène à penser que la mise au point d'une approche unificatrice et aisément transposable à chaque domaine d'application permettra d'améliorer significativement l'interopérabilité et d'en diminuer les coûts associés.

Dans le cadre de nos activités de recherche et développement, nous cherchons à mettre en œuvre une application concrète des modèles formels de partage de connaissance et choisissons les ontologies comme support de cette formalisation. Cette réalisation pose les bases d'un cadre général pour le support de l'interopérabilité fonctionnelle au niveau sémantique dans les architectures orientées services : elle a pour but de permettre l'interopérabilité des composants dans les **environnements SOA (Service Oriented Architecture) hétérogènes, distribués et hautement dynamiques**. Dans ces architectures, les services Web peuvent apparaître et disparaître à tout moment pendant l'exécution.

Pour ce type de systèmes, un niveau élevé d'interopérabilité entre producteurs et consommateurs de services est requis car les décisions de liaison ne peuvent être prises avant le déploiement ou l'exécution du système d'information. En effet, les services sont majoritairement découverts dynamiquement, à l'exécution.

Ce genre de caractéristiques pourrait se retrouver dans les systèmes d'intégration et d'information à visées militaires (Systèmes Terre et Interarmées) et civiles développées par Thales. Par exemple :

- Le besoin, sur le terrain, de déployer un système de commande et contrôle partagé par les différents alliés d'une coalition internationale (par ex. OTAN). Dans ce type de système de commande, les processus métiers correspondent à des procédures militaires qui pourraient être définis à l'avance et s'adapter à l'éventuelle indisponibilité des services utilisés.
- La nécessité, dans les systèmes civils de gestion de crise, de découvrir et coordonner les secours dans un temps contraint et de choisir les bons scénarios d'action en fonction des équipements disponibles sur le réseau.

En fonction de ces contraintes, notre cadre de travail est le suivant :

- Les SOA : de ce fait, nous manipulerons des services Web et des processus Web.
- Des domaines « métiers » spécifiés formellement : une certaine connaissance du domaine sera partagée sous forme d'ontologie(s) par les protagonistes du système.
- Des environnements hétérogènes : des différences pourront survenir aux niveaux sémantique et syntaxique entre la demande de service du consommateur et la déclaration du producteur de service.
- Des environnements dynamiques : les consommateurs de service, sous forme de processus métiers, seront liés dynamiquement lors de l'orchestration aux services disponibles dans le système.

Le reste de cet article s'organise comme suit : dans un premier temps nous nous présenterons la solution logicielle développée par Thales, ensuite nous nous attarderons sur les concepts fondamentaux traités par cette solution, nommément : la spécification des connaissances, des propriétés fonctionnelles, la publication et la recherche sémantique de service, ainsi que l'orchestration sémantique de services. Enfin, avant de conclure, nous aborderons les travaux de recherche actuellement en cours et qui s'inscrivent dans la continuité de cette réalisation.

2. Le framework SETHA

Afin d'apporter une solution simple et efficace au problème de l'interopérabilité dans les architectures SOA hétérogènes et dynamiques, nous avons mis au point un ensemble de composants réunis au sein d'un framework nommé SETHA (SEmantic THAles). Cette réalisation industrielle regroupe un ensemble extensible de fonctionnalités et technologies pour permettre :

- **La spécification** :
  - des connaissances sous forme d'ontologies (cf. 3.),
  - des propriétés fonctionnelles des services Web à partir de leur déclaration de service et d'ontologies (cf. 4.1),
  - des contraintes fonctionnelles des processus Web par une méthodologie similaire à celle employée pour les services (cf. 4.2).
- **La publication et la découverte sémantique** de services via un annuaire (cf. 5.).
- **L'orchestration sémantique** des processus Web à partir des informations syntaxiques et sémantiques disponibles au moment de l'exécution des processus (cf. 5.). Cela inclut :

- L'appariement entre consommateurs de services et les meilleures offres des producteurs grâce à la notion de conformité sémantique.
- La gestion de l'adaptation de données nécessaire à l'appel de services.

La particularité de ce framework réside donc dans ses facultés d'abstraction des contraintes syntaxiques pour se focaliser sur le sens réel des informations exprimées par les composants et utilisateurs du système.

Ces informations sémantiques permettent de définir un système d'information par composition sans avoir à se soucier des contraintes telles que l'adresse des services à appeler, le nom des opérations, le type des données : cette abstraction est d'autant plus importante, qu'obtenir une correspondance syntaxique parfaite entre offre et demande est très improbable dans un système hétérogène.

Ce framework est essentiellement constitué de technologies standardisées, ou en cours de standardisation, et issues de travaux relatifs au Web Sémantique : comme le langage SAWSDL (Semantic Annotation for WSDL), BPEL (Business Process Execution Language), OWL (Ontology Web Language) et la spécification d'annuaires de service UDDI (Universal Description, Discovery and Integration). Ceci devrait assurer la pérennité du développement et de ses futures mises à jour. Nous verrons dans les sections suivantes comment ces technologies sont misent en œuvre pour obtenir les fonctionnalités souhaitées.

3. Spécification des connaissances

Lors de la mise au point d'un système d'information reposant sur une architecture SOA, il n'existe traditionnellement pas de modèle commun des connaissances portant sur le domaine d'application du système. L'entente entre les différents intervenants se situe alors au niveau syntaxique et ne fait l'objet d'aucune réflexion poussée avant son déploiement : un client ne peut qu'extrapoler le fonctionnement effectif d'un service à partir de la description syntaxique de son interface (par ex. le nom des opérations ou le type des paramètres). Mais producteurs et consommateurs de services attribuent-ils la même signification aux lexèmes qu'ils utilisent ?

3.1. Une approche formelle : les ontologies

Cet état de fait constitue un frein majeur à l'adoption des architectures SOA dans les environnements à forte hétérogénéité. En effet, comment mener à bien des processus Web complexes si l'on n'est pas en mesure de sélectionner de manière

effective les fonctionnalités nécessaires à leur exécution parmi le vaste ensemble de services offerts par des tiers ?

C'est pour répondre à cette problématique que le framework que nous avons réalisé supporte la spécification formelle des concepts significatifs dans un ou plusieurs domaines d'application. Pour s'assurer qu'il n'existe pas de différences d'interprétation entre fournisseurs et consommateurs de service, on leur demande de faire référence à une sémantique connue et distribuable.

En fonction de leur degré de formalisation, les méthodes formelles peuvent être utilisées à des fins diverses :

- Pour spécifier les propriétés d'un système. Une méthode formelle peut donc être utilisée pour décrire les propriétés fonctionnelles des composants d'une architecture SOA (cf. 4.)
- Pour prouver que certaines propriétés sont valides dans le système.
- Pour raisonner sur les connaissances et d'effectuer des calculs d'inférence. On peut alors effectuer automatiquement certains types de raisonnement sur les propriétés fonctionnelles, par exemple pour calculer les correspondances entre offres et demandes de service (cf. 5.1.).

Dans le cadre de notre framework, cette spécification formelle sera effectuée au moyen d'ontologies. Une ontologie est un modèle des entités et relations dans un domaine spécifique ou « universe of discourse » (UoD). Elle se distingue d'une taxonomie (connaissances avec une hiérarchisation minimale ou une structure parent-enfant), d'un thésaurus (mots et synonymes) dans la mesure où elle représente un modèle conceptuel (avec des connaissances plus complexes) voir même une théorie logique. Une ontologie dispose d'une sémantique formelle, c'est-à-dire une théorie de modèle pour son langage. De ce fait, elle supporte l'inférence via son modèle formel, et peut être est décidable et soluble en fonction de son expressivité.

3.2. Un langage de spécification d'ontologies : OWL

OWL (DL McGuinness *et al.*, 2004) est le langage de spécification d'ontologies que nous avons retenu dans notre framework. Il fournit les moyens pour définir des ontologies Web structurées. Ce langage est basé sur les recherches effectuées dans le domaine des logiques de description. De plus, une description OWL d'ontologie présente l'avantage d'être « sérialisable » sous une forme XML.

Il existe trois variantes du langage OWL à l'expressivité croissante : *lite*, *DL* et *full*. OWL-DL à l'avantage de rester décidable tout en présentant une expressivité suffisamment étendue pour la plupart des applications, c'est donc lui que nous utiliserons pour la définition d'ontologies. De plus il existe de nombreux

raisonneurs logiques capables de traiter cette classe d'ontologies (B Parsia *et al.*,2004)(Jang *et al.*, 2004)(Bechhofer, 2003).

OWL est adéquat pour les travaux relatifs au « Web sémantique », car il offre une syntaxe définie strictement, une sémantique formelle et selon le niveau peut permettre des raisonnements automatisés par inférence sur les connaissances. Il est donc possible de profiter de ce format pour structurer, partager et échanger des connaissances. Il existe déjà de nombreuses ontologies modélisées à l'aide de OWL

4. Ontologies et spécification des propriétés et contraintes fonctionnelles

Il s'agit de la spécification des propriétés fonctionnelles offertes par les services Web publiés dans l'architecture SOA ainsi que de la spécification des contraintes fonctionnelles portant sur ces services et définies dans les processus Web. Dans les deux cas, les ontologies sont mises à contribution pour apporter la « connaissance » sémantique du domaine métier considéré.

Par « fonctionnel », on entend tout ce qui est directement lié au métier du service et aux fonctionnalités offertes, et non à la qualité du service rendu (temps de réponse ou de traitement, latence, …).

4.1. Offre de service, la spécification SAWSDL

Dans l'architecture que nous définissons, les services web possèdent une description syntaxique et sémantique de leurs propriétés fonctionnelles (les fonctionnalités exposées par le service) :

- La description syntaxique regroupe les informations telles que les noms d'opération, types de données, protocoles réseau et point d'accès.
- La description sémantique augmente les descriptions de service avec des concepts extraits d'une ontologie afin d'en préciser le sens.

De ce fait, ce framework est à notre connaissance l'une des premières réalisations industrielles à présenter une application concrète de la recommandation W3C (« Candidate Recommendation » précisément) SAWSDL (Lausen *et al.*, 2007). Cette extension de WSDL 2.0 permet à un fournisseur de service de définir des déclarations améliorées sémantiquement qui viennent s'ajouter au niveau syntaxique d'une spécification WSDL classique.

SAWSDL permet l'annotation de certains éléments d'une déclaration de service par des concepts sémantiques référencés grâce à une URL unique. Dans le cadre de notre réalisation industrielle, nous avons choisi les ontologies OWL comme moyen de définition de ces concepts sémantiques, les URL désigneront donc des classes

ontologiques. Ainsi, SAWSDL nous permet de tracer un lien direct entre une déclaration de service et la spécification sémantique du domaine considéré.

4.2. Processus Web, le langage BPEL

Afin de permettre la composition des fonctionnalités offertes par les services Web sous forme de processus métiers utiles dans le système d'information, nous nous basons sur la spécification BPEL (Bolie *et al.*, 2006).

Un processus BPEL permet la spécification de « macro-services » qui seront assemblés au moment de l'exécution du processus à partir des services disponibles dans l'annuaire (c'est ce qu'on appelle l'orchestration de services). Le moteur d'exécution de processus Web ActiveBPEL™ Engine[1] a été retenu pour la disponibilité de son code source, son extensibilité et la disponibilité d'un éditeur graphique.

En tant que consommateurs des services disponibles, les processus BPEL bénéficient du même type d'amélioration sémantique que les déclarations de service au format WSDL. De fait, lorsque l'on exprime une demande de service dans la spécification BPEL, celle-ci est effectuée au sein d'un fichier WSDL attenant. Dans ce cas, la déclaration WSDL ne correspond plus à une offre de service mais bien à une demande. Ainsi, en remplaçant la demande au format WSDL par un fichier SAWSDL, nous sommes en mesure d'adjoindre au processus BPEL le même type d'annotations sémantiques que pour les services.

5. Publication, recherche et orchestration sémantique de services

Les services Web correspondent à des entités dynamiques de notre architecture : ils peuvent devenir disponibles à tout moment au cours de l'exécution du système et parallèlement être découverts dynamiquement par leurs clients (ce sont les processus Web, cf. 4.2). Fournisseur et consommateurs de services doivent donc disposer d'un moyen commun et fiable pour effectuer la publication et la recherche de services.

Dans notre architecture, ces actions sont effectuées de manière centralisée par le biais d'un annuaire. Ce dernier implémente la spécification UDDI d'un annuaire de services multi-usages (Walsh, 2002). UDDI est une recommandation OASIS qui permet aux clients des services d'effectuer des recherches sur les déclarations de services Web publiées dans un annuaire donné (privés ou publiques) et aux développeurs de publier leurs services en spécifiant toute information relative à leurs interfaces (opérations, pré-requis, conformité à une spécification). Nous avons

1. Site officiel : http://www.active-endpoints.com/

retenu l'implémentation jUDDI[2] de la fondation Apache pour sa faible charge réseau, sa facilité de déploiement et son adoption dans le milieu industriel.

Nous utilisons l'annuaire jUDDI pour stocker les informations non-seulement syntaxiques, mais aussi sémantiques, liées aux services disponibles sur le réseau (d'une entreprise, d'un champ de bataille, …). Ces informations sémantiques étant alors exprimées par le biais de déclarations de services au format SAWSDL et d'ontologies de référence liées en partie au domaine d'application du service.

Il s'avère malheureusement que la spécification UDDI ne prévoit aucune facilité pour le stockage d'informations sémantiques dans l'annuaire. Pour pallier cette déficience, nous avons utilisé les capacités d'extensions du modèle de données UDDI pour mettre au point une correspondance (ou « mapping ») entre les déclarations au format SAWSDL et les structures de données de l'annuaire. Écrire une correspondance entre un domaine A et un domaine B signifie alors que l'on définit un procédé pour représenter toutes les structures de donnée du domaine A dans les structures de donnée du domaine B. Il doit être aussi possible de reconstituer tout ou partie des données de type A à partir des données mappées dans le type B.

Cette correspondance va permettre d'effectuer la transition d'une gestion des services basée uniquement sur la syntaxe et mise en œuvre avec WSDL et UDDI à une gestion basée principalement sur la sémantique et mise en œuvre grâce à SAWSDL et une couche de compatibilité sémantique pour UDDI. Du point de vue de l'implémentation de notre framework SETHA, cette couche de compatibilité correspond à la programmation d'une interface spécifique pour la publication et la recherche sémantique de services : l'API Java LUCAS (Layer For UDDI Compatibility with Annotated Semantics).

Les fonctionnalités de recherche de services fournies par l'API LUCAS rendent alors possible l'exécution de processus Web annotés par des concepts issus d'ontologies. On parle alors d'orchestration syntaxique et sémantique de services : il s'agit du processus par lequel le moteur ActiveBPEL™ Engine va évaluer les contraintes d'un processus web donné et sélectionner parmi les services Web disponibles ceux capables au mieux de mener à bien la fonctionnalité décrite par le processus, tout en gérant l'interaction entre ces services.

L'orchestration implique donc un filtrage effectif des services disponibles en fonction des contraintes syntaxiques et sémantiques exprimées dans le processus BPEL et des propriétés syntaxiques et sémantiques offertes par les services Web publiés dans l'annuaire. Pour se faire, l'orchestration de services implémentée dans notre plate-forme exploite les informations contenues dans les ontologies liées aux concepts apposés sur les déclarations de services et les descriptions de processus pour déterminer au mieux la correspondance entre contraintes et propriétés.

---

2. Site officiel : http://ws.apache.org/juddi/

5.1. Interopérabilité - la notion de conformité sémantique

Dans notre framework, une demande et une déclaration de service vont être mis en correspondance lors de l'orchestration si elles peuvent être considérées comme sémantiquement équivalentes. Le calcul de conformité nécessaire à la mise en correspondance repose sur les ontologies liées aux concepts manipulés et peut être effectué de plusieurs manières. Nous avons développé le framework de façon à ce qu'il puisse être étendu pour accepter différents calculs d'équivalence et fournissons une implémentation de base.

L'algorithme utilisé pour cette implémentation simplifiée possède une faible complexité et base le calcul d'équivalence sur les relations OWL-DL 'is-a' et 'is-equivalent-to' entre classes dans les ontologies à héritage simple. Il ne considère pas les déclarations de service dans leur intégralité mais s'intéresse plutôt aux opérations qui y sont déclarées.

Une opération est considérée comme un vecteur d'information sémantique. Soit V_{Req} une demande de service portant sur une opération et V_{Dec} une déclaration d'opération où le *goal*, les entrées et les sorties sont des concepts ontologiques, nous avons :

$V_{Req} = <goal_{Req}, inputs_{Req}, output_{Req}>$

$V_{Dec} = <goal_{Dec}, inputs_{Dec}, output_{Dec}>$

V_{Req} et V_{Dec} sont alors considérés comme équivalents si $goal_{Req}$ est un sous-concept de, ou est équivalent à $goal_{Dec}$; si chaque élément de $inputs_{Req}$ est un sous-concept de, ou est équivalent à un élément de $inputs_{Dec}$; et si $output_{Req}$ est un sur-concept de, ou est équivalent à $output_{Dec}$.

Pour illustrer l'implémentation de ce raisonnement, considérons l'exemple ci-dessous où les opérations sont décrites en utilisant des classes extraites d'une d'ontologie (cf. Figure 1). Seule une méthode est retenue car c'est la seule qui valide la précédente formule.

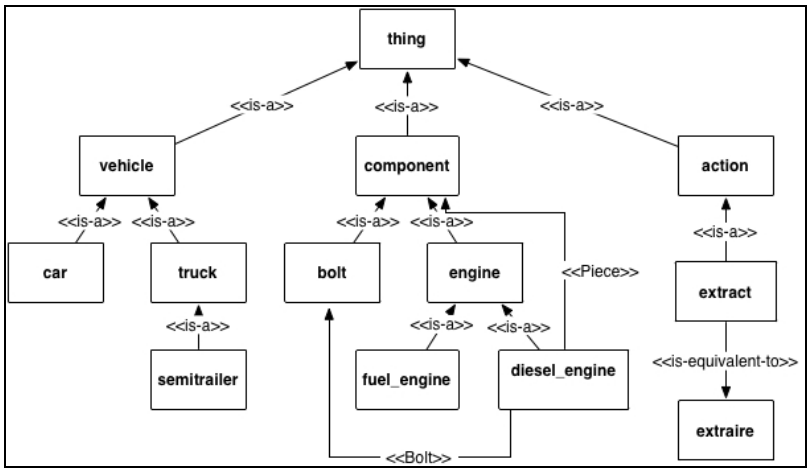

Figure 1. *Ontologie d'exemple*

Opération demandée	**Opérations disponibles sur les services**	**Equivalence sémantique**	**Commentaire**
extract(truck):engine	extraire (vehicle):diesel_engine	OK	
	extract (**semitrailer**):engine	NOK	Le concept d'entrée du service est trop spécifique par rapport au concept d'entrée demandé. Il y a perte d'information.
	extract (vehicle):**component**	NOK	Le concept de sortie du service est trop général par rapport au concept de sortie demandé. Il y a, la aussi, perte d'information.

Il est important de noter que d'autres techniques de raisonnement pourront être développées de manière à passer outre les restrictions de l'implémentation actuelle. Par exemple l'utilisation d'un raisonneur sur logiques de description permettrait, via

le mécanisme de subsomption (Sycara *et al.*, 2002) (Paolucci *et al.*, 2002), d'inférer des relations « is-a » et « is-equivalent-to » autres que celles indiqués explicitement dans l'ontologie et donc d'étendre le champ d'action du calcul de correspondance entre demande et déclaration de service.

5.2. Ontologies comme modèles de données pivot

L'utilisation d'une (ou plusieurs) ontologie(s) partagée(s) entre les différents protagonistes du système permet de s'abstraire des considérations purement syntaxiques lors de la recherche de services. Mais il ne faut pas perdre de vue qu'au moment de l'appel des services sélectionnés sémantiquement, le consommateur devra respecter le type concret des paramètres des méthodes des services ainsi que s'accommoder du type de retour.

Si les sections précédentes se sont attelées à apporter une solution au problème de l'hétérogénéité sémantique, il faut donc aussi être en mesure de la traiter au niveau syntaxique. La spécification SAWSDL que nous avons retenu dans notre framework prévoit justement l'indication d'un *lifting* et *lowering schema-mapping* pour chaque type concret de données XML Schema référencé par une déclaration SAWSDL (cf. Figure 2). Le but de ces indications supplémentaires est d'assurer la conversion entre les types utilisés par les consommateurs de service et ceux indiqués par les services eux-mêmes.

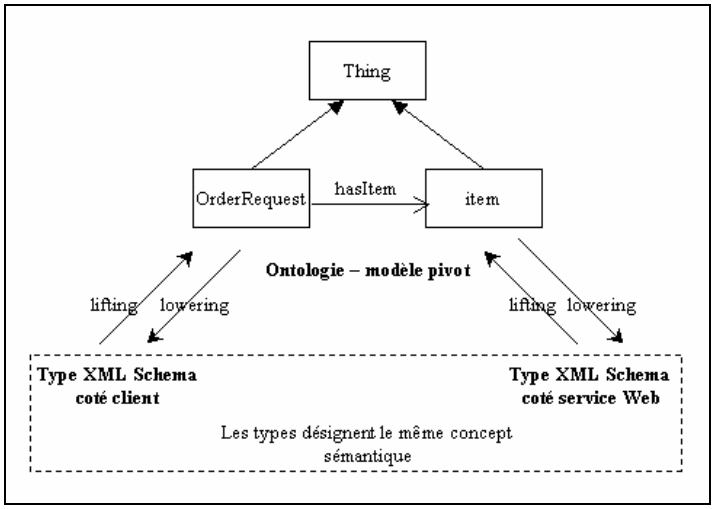

Figure 2. *Adaptation de données*

Cette conversion repose sur la conformité sémantique des différents types de données : on peut supposer que si deux types XML Schema différents sont annotés

par la même classe ontologique alors ils désignent le même concept dans le domaine considéré. En ce sens, ils véhiculent des informations similaires et **l'on peut utiliser l'ontologie du domaine comme un modèle pivot pour la conversion des données**. L'annotation *lifting schema-mapping* pointe alors vers une feuille XSLT indiquant la transformation du type XML Schema vers la structure de données de l'ontologie. *Lowering schema-mapping* effectue la conversion inverse. On est ainsi en mesure de mener à bien un appel concret de service et d'en exploiter le résultat.

6. Implémentation du framework SETHA

Toutes les fonctionnalités présentées dans cet article sont implémentées au sein d'un même framework. Elles sont réparties dans plusieurs de ses composants vitaux sous la forme de solutions existantes qui ont pu être adaptées à nos besoins ou de développements internes. On distingue :

- **L'annuaire de services Web jUDDI** : il rend visible les déclarations de services aux clients en stockant les informations syntaxiques et sémantiques relatives aux interfaces des services Web. Son utilisation dans notre framework passe par l'API LUCAS.
- **L'API LUCAS (Layer For UDDI Compatibility with Annotated Semantics)** : développé par nos soins, elle regroupe les interfaces pour la publication et la recherche sémantique de services dans l'annuaire UDDI. Dans la version actuelle du framework, c'est elle qui intègre le raisonneur logique sur ontologie qui permet de déterminer la conformité sémantique (cf. 5.1).
- **Le moteur BPEL** : ActiveBPEL™ Engine assure l'évaluation et l'orchestration des processus BPEL. Il a été modifié pour déléguer au Semantic Call Translator (SCT) la recherche sémantique de services.
- **Le SCT (Semantic Call Translator)** : développé par nos soins, il reçoit les appels de services sémantiques du moteur BPEL et va utiliser la bibliothèque LUCAS pour la recherche de services dans l'annuaire. Il gère aussi l'adaptation de données (cf. 5.2).
- **L'API Woden**[3] de la fondation Apache : pour la manipulation de fichiers au format WSDL 2.0 (SAWSDL est au format WSDL 2.0). Au moment du la mise au point du framework il s'agissait de la seule API capable d'assurer cette fonctionnalité.

Tous ces composants sont amenés à communiquer au sein de SETHA. Afin d'illustrer cette collaboration entre les différents composants, on présente deux scénarios très simples et leurs diagrammes de séquence associés.

3. Site officiel : http://incubator.apache.org/woden/

Le premier scénario correspond à la publication d'une déclaration de service suivie de la recherche d'un service capable d'assurer une opération décrite sémantiquement (cf. Figure 3).

La publication se traduit par l'appel d'une méthode *publish(sawsdl_file)* sur l'API LUCAS, où *sawsdl_file* correspond à l'url d'une déclaration de service au format SAWSDL disponible sur un serveur web (*Web Server*). Afin de charger cette déclaration en mémoire, LUCAS utilise l'API Woden qui permet d'obtenir une représentation arborescente (*sawsdl tree*) du contenu de la déclaration. Cette phase de chargement est suivie de l'application de la correspondance (mapping) vers les structures de données UDDI et la publication des éléments obtenus par cette correspondance dans l'annuaire *jUDDI* via son API standard d'accès à distance.

Nous présentons aussi la recherche d'une opération décrite sémantiquement par un concept appelé *Goal* qui précise le rôle métier accordé par le développeur du service à cette opération. La recherche de service reste ainsi complètement indépendante de la concrétisation syntaxique de l'opération sur un service Web (son nom et plus généralement sa signature). Elle est effectuée de nouveau par l'application de la correspondance (*maping(Operation, Goal)*) qui permet cette fois d'obtenir une structure de données UDDI précise (un *tModel*) que l'on recherchera dans l'annuaire *jUDDI*. Si une opération directement correspondante à été trouvée dans l'annuaire on retransmet les informations nécessaires à son appel au client de l'API *LUCAS*.

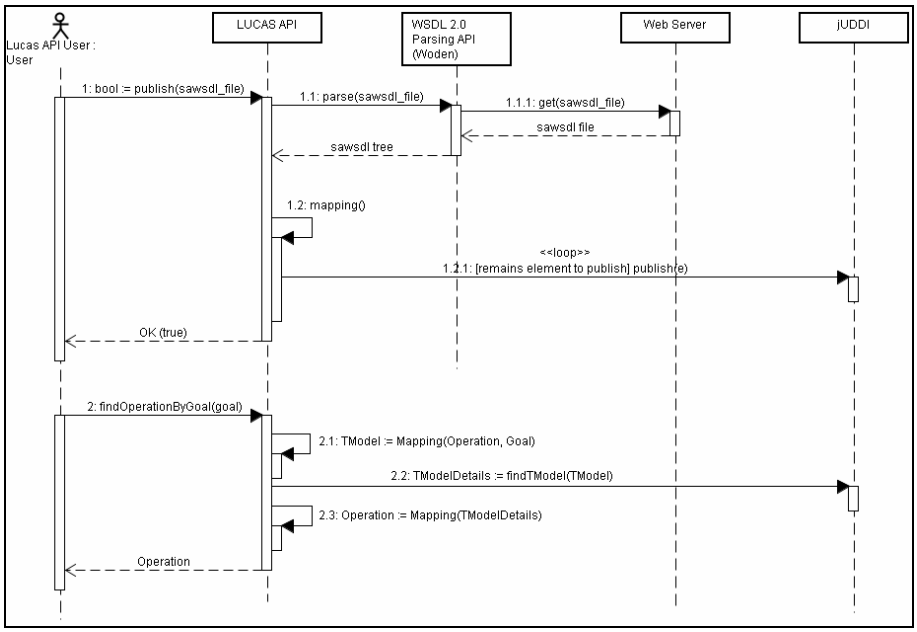

Figure 3. *Publication et recherche de service avec LUCAS*

Le deuxième scénario concerne la problématique plus générale de gestion dans le framework SETHA d'un appel indirect d'opération à partir d'un processus en cours d'évaluation par le moteur BPEL (cf. Figure 4). La signature exacte, le nom de l'opération ou son emplacement physique (url du service Web) n'ont pas besoin d'être connus dans le processus : cette fonctionnalité repose sur la recherche sémantique de service présentée ci dessus. Un appel est alors constitué d'informations sémantiques décrivant l'opération recherchée (des classes ontologiques : *goal*, *input*, *output*), d'informations syntaxiques sur les types utilisés dans le processus (pour l'adaptation de données), et de données (les valeurs de paramètres de l'opération appelée).

On peut voir que le moteur BPEL sous-traite cet appel au *SCT* qui se charge d'appliquer l'algorithme de correspondance sur ontologies (*Reasoner*) si une opération directement correspondante à l'appel initial ne peut être trouvée (cf 5.1). Une fois l'opération trouvée, c'est aussi le *SCT* qui se charge d'effectuer l'appel effectif de service tout en gérant l'éventuelle adaptation de données nécessaire entre le processus BPEL et le service Web (*Service*) sélectionné.

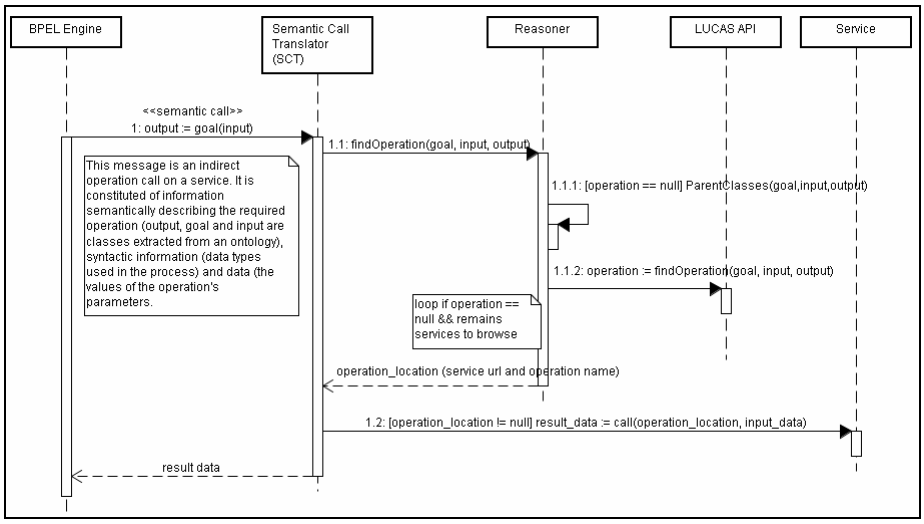

Figure 4. *Gestion bout à bout d'un appel sémantique de service*

7. Travaux en cours : gestion des contraintes non-fonctionnelles

Afin de se rapprocher des contraintes présentes lors du déploiement d'un système d'information hétérogène et dynamique, le travail présenté dans cet article se doit d'être étendu au-delà des considérations fonctionnelles.

En effet, nous devons ajouter la gestion des contraintes non-fonctionnelles (telles que la Qualité de Service, QoS) dans le framework. Ce sujet est actuellement en cours d'investigation à Thales Communications dans le cadre d'une thèse de doctorat. Nous cherchons ainsi à établir une orchestration dynamique de service qui ne tienne plus seulement compte des propriétés fonctionnelles des services (aux niveaux sémantiques et syntaxiques) mais qui soit aussi basée sur des contraintes non-fonctionnelles et sur des valeurs de QoS instantanées.

Dans l'état actuel de ces travaux, nous cherchons à mettre en place un framework extensible capable de supporter différents types de raisonnement sur les propriétés fonctionnelles (définies statiquement) et sur les propriétés non-fonctionnelles définies statiquement mais aussi dynamiquement par le biais de valeurs instantanées de QoS sur les services.

Un autre aspect de ces travaux concerne la définition de modèles de décision effectifs sur les aspects fonctionnels et non-fonctionnels : plus particulièrement la décision de liaison d'une demande à une offre de service au moment de l'orchestration. Différentes méthodologies pourraient être utilisées, telles que la

décision multicritères ou les processus de décision markoviens commandés, afin de mener à bien et optimiser cette décision.

8. Travaux connexes

La plupart des travaux autour de la découverte et l'orchestration de services Web sémantiques ne traitent qu'un sous-ensemble des problématiques abordées par ce framework. Ils se focalisent sur la phase d'appariement sémantique entre offre et demande de service mais ne considèrent pas les étapes supplémentaires qui sont nécessaires au bon déroulement d'un appel de service : la gestion des contraintes non-fonctionnelles (cf. 7.), l'adaptation des données échangées entre consommateurs et producteurs (cf. 5.2) ou encore les cas de non-correspondance sémantique directe entre offres et demandes (cf. 5.1).

Les travaux suivants supposent la disponibilité d'ontologies fonctionnelles pour décrire les « capacités » des services mais préconisent différentes approches pour l'appariement. Pour certains cela se résume à calculer la relation de subsomption directe entre offre et demande: c'est par exemple le cas dans (Li *et al.*, 2004), où les auteurs présentent l'implémentation d'un prototype utilisant une ontologie DAML-S et un raisonneur logique à cette fin. Ils basent le calcul d'équivalence sur la notion de subsomption en Logique de Description mais ne traitent pas l'adaptation de donnée après l'appariement.

Pour d'autre, les fonctionnalités des services sont décrites par rapport aux transformations d'état qu'elles génèrent. La comparaison entre offre et demande repose alors sur les concepts sémantiques utilisés pour décrire les entrées et sorties des services. C'est cette seconde approche qui à été généralisée dans SETHA. La subsomption est, là aussi, mise à contribution, mais pas au même niveau d'abstraction que dans les travaux précédents :

- Dans (Paolucci et al., 2002), c'est DAML-S qui à été choisi pour décrire les services. Les auteurs abordent la problématique du calcul de correspondance entre offre et demande de service du point de vue de leur description sémantique, mais donnent peu d'information quant à leur implémentation. Contrairement aux autres articles présentés dans cette section, Paolucci et al. ont souhaité conserver l'annuaire UDDI pour la recherche et la publication de services, ils rejoignent notre approche sur ce point.
- (Di Noia et al., 2003) ont mis au point un processus d'appariement reposant sur la subsomption entre concepts en Logique de Description. Ils s'attachent spécifiquement à la définition de l'algorithme de calcul mais n'abordent pas les autres problématiques des SOA sémantiques. Les auteurs distinguent trois niveaux de correspondance entre offre et demande : totale, potentielle et partielle.

- Dans (Sycara *et al.*, 2002), les auteurs présentent le langage LARKS de spécification d'offre et demande de service ainsi qu'un processus d'appariement qui traite les aspects syntaxiques et sémantiques. Ces travaux utilisent eux aussi les ontologies (langage ITL) et la subsomption mais ne sont pas liées spécifiquement aux SOA. D'ailleurs, contrairement à SAWSDL, une offre de service ne précise pas les informations réseau (protocole, URL du composant,…).

9. Conclusion

Dans son état actuel, cette implémentation constitue la première étape vers la réalisation d'un framework complet de support des SOA sémantiques qui pourrait être réutilisé au sein des activités civiles et militaires du groupe Thales. À maturation, il aurait pour vocation d'être adopté par le plus grand nombre à l'intérieur et à l'extérieur du groupe. Les choix technologiques relatifs à cette plate-forme ont été faits dans ce sens : nous favorisons les solutions libres (« *open-source* ») et standardisées.

Mais, comme nous avons pu le voir, certains points demandent à être approfondis, notamment dans le domaine de l'adaptation des données et du raisonnement sur ontologies : les évolutions possibles passent essentiellement par la mise au point de techniques plus puissantes de raisonnement ontologique (inférence logique, alignement d'ontologies, utilisation de logiques plus évoluées comme les logiques floues ou multi-valuées) et la couverture d'un ensemble plus étendu de situations (des appariements (*matching)* plus pertinents, une adaptation de données plus robuste). Ces points précis se doivent d'être étudiés avant d'entrer en phase finale d'exploitation du framework SETHA.

Une autre possibilité d'évolution fait parallèlement l'objet de recherches : elle consiste à définir un framework générique de gestion des contraintes fonctionnelles et non-fonctionnelles (telles que la Qualité de Service, QoS) dans les SOA. Le principe étant de garder une approche extensible capable de supporter différents types de raisonnement interchangeables et complémentaires sur ces propriétés et contraintes tout en s'inspirant des travaux déjà réalisés dans le domaine.

10. Remerciements

Jacques Malenfant – Laboratoire d'Informatique de Paris VI (LIP6)

Jacques.Malenfant@lip6.fr

Université Pierre et Marie Curie

104, avenue du Président Kennedy

F-75016 Paris

Hugues Vincent – Thales Communications France

hugues.vincent@fr.thalesgroup.com

1 à 5 avenue Carnot

91883 Massy Cedex - France

11. Bibliographie

Bechhofer S., *Hoolet OWL Reasoner*, 2003, http://owl.man.ac.uk/hoolet/

Bolie J., Cardella M., Blanvalet S., Juric M., Carey S., Chandran P., Coene Y., Geminiuc K., Zirn M. & Gaur, H. *BPEL Cookbook: Best Practices for SOA-based integration and composite applications development Packt Publishing*, 2006.

Di Noia T., Di Sciascio E., Donini F., Mongiello M., *Semantic matchmaking in a P-2-P electronic marketplace*, Proc. Symposium on Applied Computing (SAC'03), 2003, 582-586

Jang M., Sohn J., *Bossam: An Extended Rule Engine for OWL Inferencing Workshop on Rules and Rule Markup Languages for the Semantic Web* at the 3rd International Semantic Web Conference (LNCS 3323), Springer, 2004, 128-138.

Lausen H., Innsbruck D., *Semantic Annotations for WSDL and XML Schema* 2007.

Li L., Horrocks I., *A Software Framework for Matchmaking Based on Semantic Web*, Technology International Journal of Electronic Commerce, ME Sharpe, 2004, 8, 39-60

McGuinness DL., van Harmelen F. et al, *OWL Web Ontology Language Overview*, W3C Recommendation, 2004.

Paolucci M., Kawamura T., Payne T., Sycara K., *Semantic Matching of Web Services Capabilities*, Proceedings of the 1st International Semantic Web Conference (ISWC), Springer, 2002, 348

Parsia B., Sirin E., *Pellet: An OWL DL Reasoner*, Proceedings of the International Workshop on Description Logics, 2004.

Sycara K., Widoff S., Klusch M., Lu J., *Larks : Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace Autonomous Agents and Multi-Agent Systems*, Springer, 2002, 5, 173-203

Walsh A., *Uddi, Soap, and Wsdl: The Web Services Specification Reference Book* Prentice Hall Professional Technical Reference, 2002.