MajecSTIC 2009
Avignon, France, du 16 au 18 novembre 2009

Méthodologie pour l'orchestration sémantique de services dans le domaine de la fouille de documents multimédia

Jérémie Doucy[1 2], Habib Abdulrad[1], Patrick Giroux[2], Jean-Philippe Kotowicz[1]

1 : INSA de Rouen, Laboratoire LITIS - EA 4108, Avenue de l'Université - BP 8, 76801 Saint-Étienne-du-Rouvray - France.
2 : EADS Defence & Security, Information Processing Control and Cognition, Parc d'Affaire des Portes BP 613, 27106 Val De Reuil Cedex - France.

Contact : jeremie.doucy@litislab.eu

Résumé
Cet article présente une nouvelle approche, basée sur les standards existants, pour la construction de chaînes de traitement dans le domaine de la fouille de documents multimédia. En utilisant le paradigme des architectures orientées services, l'approche que nous présentons ici permet de simplifier drastiquement la création de ces chaînes et ouvre une voie vers la validation et l'automatisation de processus « métiers » complexes.

Abstract
This paper presents a new approach, based upon existing standards, to construct multimedia processing chains. Using service-oriented architecture this approach eases chains construction and enables automatic validation and generation of processes.

Mots-clés : Orchestration, sémantique, WebLab, BPEL, SOA

Keywords: Orchestration, semantic, WebLab, BPEL, SOA

---

1. Contexte

Depuis des années, les architectures orientées services (SOA) [10] se sont imposées comme une réponse efficace aux problèmes d'intégration posés par la multiplicité des composants logiciels nécessaires au développement d'applications modernes.
L'utilisation du paradigme SOA déplace la complexité des tâches d'intégration au niveau de la définition des différentes chaînes de traitement, ce qui s'apparente à expliciter de quelles manières sont composés les différents services. Les experts d'un domaine métier sont capables de les définir mais ils ne possèdent pas la formation nécessaire pour résoudre les problèmes d'interopérabilité technique inhérents à cette définition.
Par exemple, un voyagiste est le plus apte à définir un processus complexe de réservation de billets et d'hotels. Cependant ces processus peuvent varier relativement souvent en fonction des pays de destination ou même simplement de nouvelles offres disponibles. Il est donc inévitable de redéfinir régulièrement ces processus, et ceci nécessite, pour le moment, une grande connaissance technique.

2. Problématique

Dans l'état actuel de la technologie, il est donc indispensable de recourir à la fois aux compétences d'un expert en orchestration, qui peut être assimilé à un programmeur de processus, et à celles d'un expert du domaine afin de décrire précisément les chaînes de services à mettre en place.

L'un des principaux avantages du paradigme SOA est d'offrir de fortes capacités d'évolution. En effet, le dynamisme offert par ce type d'approche permet une maintenance extrêmement simplifiée au niveau de la plateforme pour autant que l'administrateur de cette dernière maîtrise parfaitement les aspects techniques de la création de chaînes de traitement.
C'est pourquoi, lors de l'exploitation d'une plateforme SOA, la majorité des administrateurs se plaignent de la complexité d'évolution de leurs applications lors de la mise à jour ou l'ajout de nouveaux services. Ils sont, dans la plupart des cas, obligés de faire appel à des experts en langages d'orchestration afin de mettre à jour l'automatisation des processus.
En d'autres termes, si le paradigme SOA est une avancée majeure dans la définition d'architectures réparties et distribuées, où plusieurs partenaires doivent coopérer au sein d'un même tissu applicatif, il ne résout pas tous les problèmes. Il permet effectivement de simplifier et de rendre plus robuste les tâches d'intégration, de maintenance et de suivi sous réserve que le concepteur du processus métier soit parfaitement formé et compétent dans ces domaines techniques complexes. Il est donc nécessaire de trouver des méthodes afin de rendre accessible au plus grand nombre, non pas l'administration technique, mais l'administration applicative, voire opérationnelle, d'une plateforme SOA.

3. Solutions existantes

Au cours des dernières années, le couple BPEL [1] / WSDL [4] a émergé comme standard utilisé pour la description et l'orchestration de services.
WSDL permet la spécification d'interfaces et de méthodes, ainsi que de leurs paramètres en terme d'entrées, sorties et exceptions.
Basé sur cette définition de service, le standard de description des annuaires de services est UDDI [3]. Il permet de référencer des services décrits à l'aide de WSDL. Il répond aux problématiques de fournisseurs et consommateurs de services, notamment avec une gestion des droits d'accès aux services référencés dans l'annuaire.
BPEL est le langage d'exécution de chaînes le plus utilisé, implémenté et donc validé. Cependant c'est un langage de très bas niveau, écrit en XML et basé sur WSDL, il est très technique et assez difficile à prendre en main même en utilisant des éditeurs graphiques.
Ce couple est nécessaire car il existe actuellement de nombreux outils permettant de rendre la création de services web et de chaînes de traitement accessibles, comme la génération automatique de codes sources client et serveur dans pratiquement tous les langages de programmation ou encore les éditeurs graphiques évolués permettant la création efficace de chaînes BPEL.
Toutes ces technologies sont maintenant matures et résolvent les nombreux problèmes d'interopérabilité technique présents dans les plateformes SOA. En effet, le travail des architectes experts des plateformes orientées services est maintenant grandement simplifié grâce à ces standards. Cependant toutes ces technologies ne sont pas triviales et nécessitent une grande maîtrise technique afin de pouvoir en tirer profit. Les tâches d'administration classique d'une plateforme SOA restent donc réservées à des experts techniques.
La piste la plus prométeuse en vue de simplifier, voir d'automatiser, certaines tâches dans le but de les rendre accessibles à un expert non technique est l'utilisation de descripteurs sémantiques.
En effet, de nombreux travaux ont été menés dans le but d'ajouter « du sens » au niveau de la description des services afin de compléter notamment la description technique offerte par le WSDL, c'est-à-dire d'ouvrir la voie vers une interopérabilité sémantique. Complémentaire de l'interopérabilité technique, l'interopérabilité sémantique garantit principalement la cohérence des données échangées au sein d'une même plateforme et plus généralement au sein d'un domaine métier.
Elle nécessite donc l'ajout de descripteurs sémantiques à la définition de services. Dans ce domaine, on distingue trois initiatives de recherches : SAWSDL [9], WSMO [5] et OWL-S [11].
SAWSDL permet l'ajout d'annotations sémantiques au niveau d'un service (service, méthode appelée, messages échangés, ...) mais aussi au niveau d'un schéma XML (pour tout élément d'un paramètre). Il est donc possible de lier pratiquement chaque élément de la définition d'un service à une classe ontologique, autrement dit, d'ajouter du sens à tout élément de description syntaxique.
WSMO fourni une plateforme complète qui offre un cadre de développement visant à simplifier la création d'applications sémantiques. Cette plateforme définit notamment la notion de média-

teur. Certes cette plateforme a l'avantage d'avoir déjà été implémentée. Cependant elle reste assez imprécise en ce qui concerne la réalisation de chaînes de services.
OWL-S est une ontologie écrite en OWL [2] qui ajoute une description sémantique à un service ou un ensemble de services. Un service est défini à travers trois parties : *Profile*, *Process Model* et *Grounding Model*. Le *Profile* décrit les capacités du service dans le but d'en informer le consommateur du service. Le *Process Model* spécifie le comportement du service. Si le service est un service composite, il définira sa composition à l'aide du modèle des tâches OWL-S. Si le service est atomique, il spécifiera les IOPE [1]. Enfin le *Grounding Model* spécifie les aspects techniques du service et notamment les points nécessaires à son appel à proprement parler. La plupart du temps il s'agit d'un lien vers le WSDL du service.
OWL-S parait être une piste réellement intéressante. Cependant ce standard n'a pas encore, à ce jour, été implémenté pour la partie définition de chaînes de services.
Pour éviter d'introduire de nouveaux langages, et donc le développement de nouveaux outils, nous proposons une méthode qui se base sur le couple BPEL / WSDL mais qui tire parti des premiers résultats des recherches en cours dans le domaine de la définition sémantique de services.

4. Notre solution

Pour répondre aux besoins de simplifications lors de la création de chaînes de traitements au sein de plateformes SOA, nous avons choisi une méthodologie basée sur les standards existants. En effet, notre objectif de recherche n'étant en aucun cas de redéfinir ou de redévelopper une plateforme complète, il est donc nécessaire de s'appuyer sur les solutions actuellement utilisées.
La première étape dans la réalisation d'une plateforme SOA capable de réconcilier les experts techniques et les experts du domaine est de solutionner le problème de l'interopérabilité technique. Parmi les solutions existantes, les ESB [2] apparaîssent comme une solution intéressante. En effet ces plateformes permettent d'exposer des services définis en WSDL et cela quelque soit le protocole technique utilisé pour communiquer avec ces composants logiciels. Les ESB permettent aussi un déploiement dynamique des services au sein d'un annuaire. Cette technologie permet donc de résoudre les problèmes d'interopérabilité technique tout en préservant le couple WSDL / BPEL déjà connu des experts.
En ce qui concerne l'interopérabilité sémantique, nous préconisons de définir des interfaces de services génériques à l'aide de WSDL. Par interfaces génériques, nous entendons des interfaces métiers, proches du domaine d'application de la plateforme. Il est important de noter qu'elles ne définissent pas uniquement les méthodes exposées par ces services mais aussi les paramètres utilisés par ces méthodes. Ces paramètres doivent être partagés au maximum entre les interfaces génériques définies pour un domaine d'application donné. En d'autres termes, il est très important d'utiliser un modèle technique « pivot » qui défini la structuration des données échangées. Ceci permet, non seulement de simplifier la définition de chaînes en proposant un nombre limité mais maîtrisé d'interfaces de services, mais évite aussi toute confusion sémantique.
L'utilisation de ces interfaces génériques permet de définir plusieurs nouveaux concepts, les chaînes de traitement génériques ou « patrons » de chaînes et les chaînes dynamiques.
En effet, comme les interfaces de services sont, dans notre cas, prédéfinies il devient possible de créer une ou plusieurs chaînes génériques en fonction du domaine applicatif utilisé. En d'autres termes, en utilisant les définitions de services génériques, il est possible de prévoir les interactions entre ces services et donc de prédéfinir des chaînes de traitement correspondant à une tâche métier.
Afin de pouvoir instancier ces patrons de chaînes, il est nécessaire de choisir dynamiquement un ou plusieurs services implémentant l'interface générique à appeler. Ce qui nous amene à la définition de chaînes non seulement génériques mais aussi dynamiques.

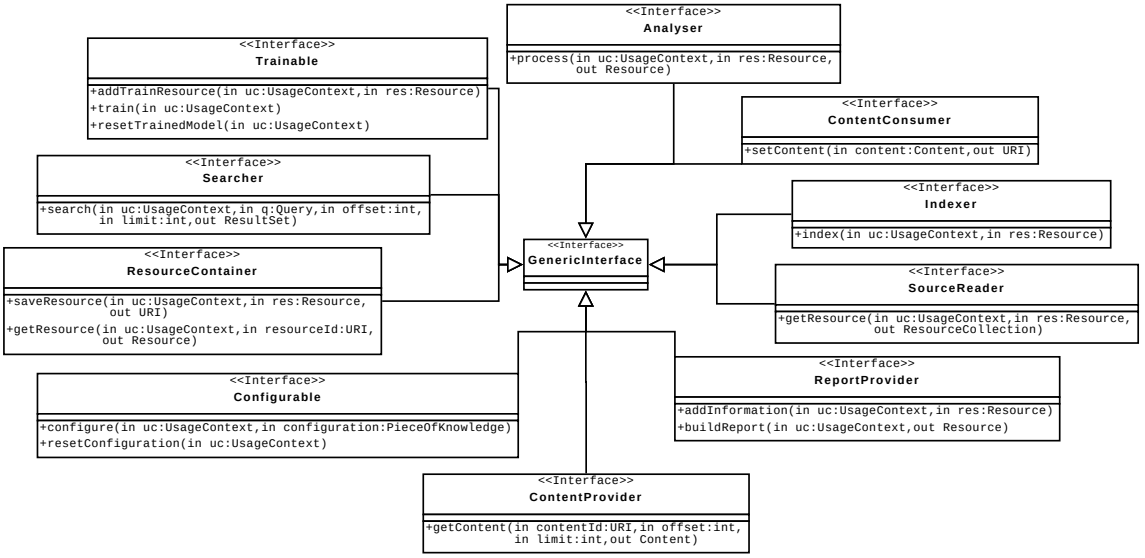

FIG. 1 – Interfaces génériques définies par la platerforme WebLab.

5. Application à la fouille de documents multimédia

Nous avons donc appliqué ces nouveaux principes à la création d'applications pour la fouille de documents multimédia. Actuellement, deux plateformes existent dans ce domaine UIMA [6] et WebLab [7]. Nous avons naturellement choisi le WebLab car cette plateforme est développée en collaboration avec notre laboratoire. Ce choix est aussi motivé par le fait que UIMA ne s'appuie pas sur les standards reconnus et semble moins souple pour la construction de chaînes. En effet, les équivalents de patrons de chaînes, pour UIMA étant prédéfinis et difficilement modifiables.
Le WebLab est basé sur le couple WSDL / BPEL en ce qui concerne la définition des interfaces génériques tout comme la création de chaînes de traitements. Cette plateforme est donc cohérente avec nos travaux. Le WebLab définit un ensemble de services génériques, explicités dans la figure 1, issue de l'expertise métier développée par notre laboratoire et ses partenaires.

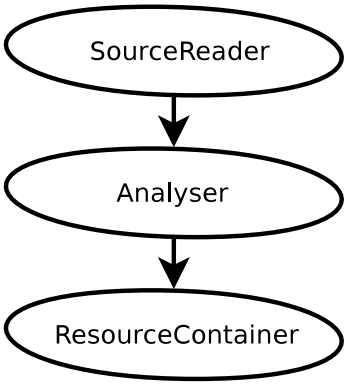

FIG. 2 – Exemple de « patron » de chaîne de traitements multimédia WebLab.

[1] Input Output Preconditions Effects
[2] Enterprise Service Bus

Fort de notre expérience en définition de chaînes pour la plateforme WebLab, nous avons défini un « patron » à l'aide des interfaces génériques prédéfinies. Nous avons relevé que la grande majorité des processus multimédia utilisés par la plateforme WebLab étaient structurés comme le montre la figure 2.
Dans la majorité des cas, une chaîne de traitements multimédia est constituée d'un service de collecte de documents (*SourceReader*), d'un service d'analyse (*Analyser*) de documents qui peut être composite et enfin d'un service de stockage de documents analysés (*ResourceContainer*).
Étant donné la nature même d'un « patron », il est nécessaire que les trois services appelés par cette chaîne soient eux-mêmes génériques, c'est-à-dire que les services utilisés par un processus défini génériquement soient abstraits.
La plateforme WebLab utilise un bus de service afin de garantir la généricité de ses services et plus particulièrement de ses chaînes de services. Chaque service déployé sur le bus est défini de façon abstraite grâce à une des *interfaces* génériques. Le lien entre cette interface et son implémentation n'est donc réalisée qu'au moment de l'exécution. En d'autres termes, lorsque l'on appelle un *endpoint*, ou identifiant de service, sur le bus, ce dernier est capable de router le message reçu vers la bonne implémentation en utilisant un protocole adapté.
Pour ce faire, le bus possède un annuaire qui est mis à jour dynamiquement et consultable en utilisant trois niveaux d'abstraction :
– Interface
– Service-name
– Endpoint
Par exemple, le tableau 1 présente la contenu d'un annuaire de service avec cinq services déployés.

Interface	*Service-name*	*Endpoint*
Analyser	NamedEntitiesExtraction	nee1
Analyser	NamedEntitiesExtraction	nee2
Analyser	LanguageDetection	lang1
Analyser	BetterNamedEntitiesExtraction	chainnee1
ResourceContainer	XMLRepo	xmlrepo1

TAB. 1 – Exemple du contenu de l'annuaire du bus

Dans ce cas si l'on demande à l'annuaire les *endpoints* qui ont pour interface ResourceContainer, il nous retournera *xmlrepo1*. Evidemment si l'on demande les *endpoints* d'interface *Analyser*, ce dernier nous renverra *nee1*, *nee2*, *lang1* et *chainnee1*. Enfin si l'on demande les *endpoints* qui ont pour *service-name NamedEntitiesExtraction* ce dernier retournera seulement *nee1* et *nee2*.
Il est intéressant de remarquer que dans cette abstraction, le bus considère une chaîne de traitement comme un service. Dans l'exemple précédent, le service *BetterNamedEntitiesExtraction* est en fait instancié à l'aide d'un moteur BPEL, mais au moment de consulter l'annuaire cette information est cachée. Seul le routeur du bus est capable de retrouver cette information et donc de lier dynamiquement l'*endpoint chainnee1* avec un appel à un processus BPEL et non un appel SOAP [3] comme pour les autres services déployés sur le bus.
Toujours dans l'optique d'utiliser au maximum les standards, les « patrons » de chaînes sont définis à l'aide de BPEL. Il est donc nécessaire de trouver un mécanisme permettant de désambiguiser chaque service du processus.
BPEL défini la notion de *partnerLink* qui, pour simplifier, permet de préciser quels services vont pouvoir être invoqués par la chaîne. Un *partnerLink* est lié à une interface donnée et simplement une interface, ce qui est l'abstraction de plus haut niveau pour le bus. Lorsque le moteur d'orchestration rencontre un element *invoke*, qui permet d'appeler un service et qui est lié à un *partnerLink*, il a connaissance de l'interface qu'il doit appeler mais en aucun cas du *service-name* et encore moins

[3] Simple Object Access Protocol : `http://www.w3.org/TR/soap/`

de l'*endpoint*. Donc, dans ce cas, le moteur demandera à l'annuaire un des *endpoints* implémentant l'interface définie dans le *partnerLink*. Imaginons un *partnerLink* lié à l'interface *Analyser* et une opération *invoke* qui est excécutée sur ce *partnerLink*, le moteur choisira donc un des services disponibles implémentant cette interface et l'appelera. Si l'on reprend l'exemple précédent, le moteur appelera un des quatres services implémentant cette interface sans que l'on puisse choisir lequel. Cependant, un *partnerLink* est une variable en BPEL et peut donc être affecté. Ceci permet de préciser à l'exécution l'*endpoint* et le *service-name* que l'on veut invoquer. Si seulement le *service-name* est précisé, le routeur du bus de services choisira un des *endpoints* déployés en utilisant ce *service-name* comme expliqué précedemment.
Cette désambiguation dynamique permet d'instancier à la volée des chaînes de traitements génériques prédéfinies. De plus, au cours de notre étude des processus de traitements multimédia existants, nous avons mis en avant la nature linéaire de la majorité des chaînes d'analyses. En effet le principe même de la plateforme WebLab, et plus particulièrement de ses chaînes, est de permettre la succession de services d'analyses complémentaires ou concurentes dans le but d'enrichir les documents traités. Cette suite est instanciée en BPEL par une succession statique d'appels à un *partnerLink* déclaré comme *Analyser*. Cette approche est satisfaisante dans un premier temps mais ne répond pas complètement au besoin de simplification de construction de processus BPEL. En effet, la mise à jour des chaînes linéaires est certes simplifiée, étant donnée que la description de ce type de processus est répétitif et cyclique, mais nécessite toujours une certaine maîtrise de BPEL.
Nous avons donc envisagé une extension de cette solution comme explicité dans la figure 3.

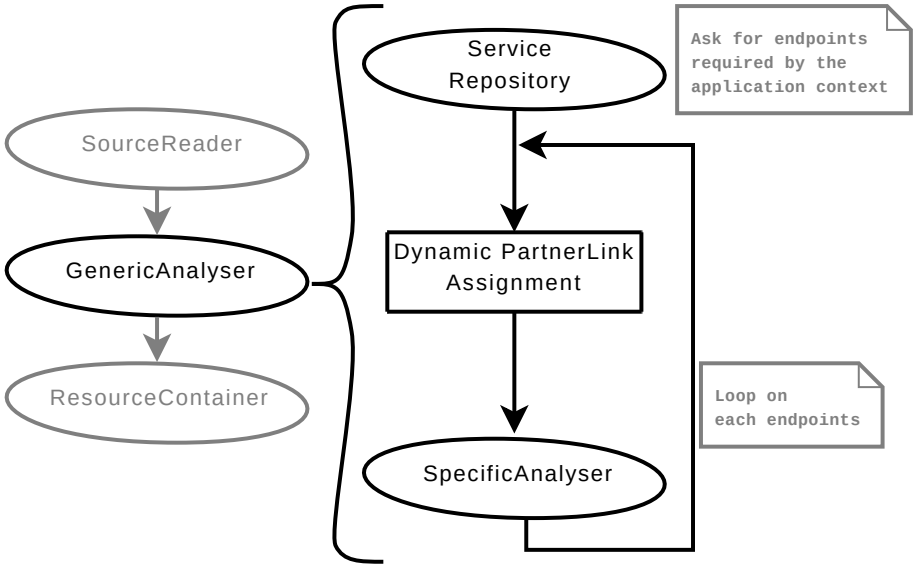

FIG. 3 – Dynamisation de la partie linéraire d'une chaîne de traitement.

Cette solution se base toujours sur le même « patron » de chaîne mais se concentre sur la dynamisation de la partie analyse du processus. Cette partie étant linéaire dans le cadre de ce « patron », la seule donnée nécessaire à l'instanciation de cette chaîne est la liste ordonnée des analyseurs à utiliser. Pour ce faire on boucle sur la liste d'*endpoints* retournés par un annuaire de service. Ensuite pour chaque analyseur récupéré, on assigne dynamiquement le *partnerLink* avant d'appeler le service que l'on vient d'identifier à l'exécution. Enfin, l'interface d'analyse prenant en paramètre et retournant une resource WebLab, on copie la réponse du service dans la requête du prochain.

6. Cas d'utilisation

Prenons l'exemple d'un administrateur qui, pour une application de traitement de données textuelles, définit une chaîne d'analyse qui doit extraire des entités nommées, géolocaliser les villes dans des textes et extraire les relations possibles entre des entités et des villes. Il s'agit donc d'un service composé de trois services atomiques : *nameEntitiesExtractor*, *geolocalisator* et *relationExtractor*.
Pour une application multimédia, il définit une autre chaîne d'analyse composée cette fois d'un extracteur de descripteurs visuels sur les images et d'un classifieur. Il s'agit alors d'une chaîne composée de deux services atomiques : *imageFeatureExtractor* et *imageClassifier*.
En utilisant un orchestrateur BPEL classique, cet administrateur doit décrire deux chaînes distinctes. Une première qui appellera séquentiellement trois *endpoints* : *nameEntitiesExtractor*, *geolocalisator* et *relationExtractor* ; et la seconde qui appellera séquentiellement deux endpoints : *imageFeatureExtractor* et *imageClassifier*.
En utilisant notre approche de « patron » combinée à un *Analyser* dynamique, l'administrateur a simplement besoin de décrire la liste des *endpoints* qui doivent être appelés pour chacune des applications. Dans un premier temps il définit donc les trois endpoints *nameEntitiesExtractor*, *geolocalisator* et *relationExtractor* pour l'application de traitement de données textes et ensuite les deux endpoints *imageFeatureExtractor* et *imageClassifier* pour l'application multimédia.
Les différents liens entre applications et endpoints sont stockés dans un annuaire de services qui est interrogé en spécifiant un identifiant d'application.
Lors de l'exécution de la première chaîne, le moteur d'orchestration fait un appel à cet annuaire en utilisant l'identifiant d'application adéquate, ce qui lui permet de récupérer la liste des *endpoints* à appeler pour cette application. Il suffit ensuite de boucler sur cette liste et d'invoquer chaque *endpoint* ainsi récupéré dynamiquement.
Cette méthode présente plusieurs avantages. Premièrement, ce n'est plus l'administrateur ou l'opérationnel qui décrit la chaîne de traitement en BPEL. Ce dernier doit simplement préciser quels services doivent être appelés et dans quel ordre sans se soucier des éventuelles manipulations de variables ou encore définitions d'interfaces. De plus, la chaîne générique utilisée est testée, validée, éprouvée ce qui évite les problèmes qui peuvent apparaître lors de définitions multiples de chaînes. Enfin lorsque la chaîne générique est mise à jour, toutes les applications sont mises à jour automatiquement.

7. Conclusions et perspectives

Le principal avantage de notre approche est de conserver le couple WSDL / BPEL ainsi que les plateformes SOA existantes tout en augmentant leur dynamisme et donc en améliorant leur administration applicative.
Cette approche de construction de chaînes ouvre de nouvelles perspectives. En effet en utilisant les dernières avancées dans le domaine du web sémantique et plus particulièrement la possibilité de définir sémantiquement un service composite ou atomique, il est possible de décrire précisemment les besoins d'une application de fouille multimédia.
Il est nécessaire d'utiliser un *annuaire sémantique* de services plutôt qu'un simple annuaire applicatif. En effet cette nouvelle approche permettra une définition plus simple, précise et efficace des chaînes de traitement en masquant complètement les aspects techniques.
Enfin, en utilisant les IOPE récupérés à l'aide des descriptions sémantiques de services, il sera possible de valider sémantiquement des instances de chaînes de traitement. On pourra, par exemple, vérifier que les préconditions d'un service correspondent bien avec les effets de son prédécesseur pour le cas des traitements linéaires. Ces IOPE permettront aussi d'assister la définition des applications en proposant des services répondants aux besoins exprimés sémantiquement par les architectes. La proposition automatique de chaînes de traitement d'après ces besoins sémantiques serait alors tout à fait envisageable.
Pour ce faire il est nécessaire de raisonner à partir des IOPE. Une des approches envisagée est la programmation logique par contraintes.
Nous avons commencé des travaux dans ce domaine en définissant un méta-modèle de processus

à l'aide du raisonneur Alloy [8] et les premiers résultats sont prometteurs.

Bibliographie

1. Charlton Barreto, Vaughn Bullard, Thomas Erl, John Evdemon, Diane Jordan, Khanderao Kand, Dieter Königand Simon Moser, Ralph Stout, Ron Ten-Hove, Ivana Trickovic, Danny van der Rijn, et Alex Yiu. Web services business process execution language version 2.0. Technical report, OASIS, May 2007.
2. S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, L.A. Stein, et al. OWL web ontology language reference. *W3C recommendation*, 10 :2006–01, 2004.
3. T. Bellwood, L. Clement, D. Ehnebuske, A. Hately, M. Hondo, Y.L. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter, et al. Uddi Version 3.0. *Published specification, Oasis*, 2002.
4. E. Christensen, F. Curbera, G. Meredith, et S. Weerawarana. Web services description language (Wsdl). *W3C Web Site*, 2001.
5. C. Feier, D. Roman, A. Polleres, J. Domingue, M. Stollberg, et D. Fensel. Towards intelligent web services : The web service modeling ontology (WSMO). In *International Conference on Intelligent Computing (ICIC)*, 2005.
6. D. Ferrucci et A. Lally. UIMA : an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4) :327–348, 2004.
7. Patrick Giroux, Stephan Brunessaux, Sylvie Brunessaux, Jérémie Doucy, Gérard Dupont, Bruno Grilheres, Yann Mombrun, et Arnaud Saval. Weblab : An integration infrastructure to ease the development of multimedia processing applications. In *International Conference on Software and System Engineering and their Applications (ICSSEA)*, 2008.
8. D. Jackson. Alloy : a lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(2) :256–290, 2002.
9. J. Kopecky, T. Vitvar, C. Bournez, et J. Farrell. Sawsdl : Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, 11(6) :60–67, 2007.
10. C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F Brown, Booz Allen Hamilton, et Rebekah Metz. Reference model for service oriented architecture 1.0. Technical report, OASIS, October 2006.
11. David Martin, Massimo Paolucci, Sheila Mcilraith, Mark Burstein, Drew Mcdermott, Deborah Mcguinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan, et Katia Sycara. Bringing semantics to web services : The owl-s approach. *Lecture Notes in Computer Science*, 3387 :26–42, 2005.