



HAL
open science

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

Jérémie Doucy, Habib Abdulrab, Patrick Giroux, Jean-Philippe Kotowicz

► **To cite this version:**

Jérémie Doucy, Habib Abdulrab, Patrick Giroux, Jean-Philippe Kotowicz. Méthodologie pour l'orchestration sémantique de services dans le domaine de la fouille de documents multimédia. 2009. hal-00437463

HAL Id: hal-00437463

<https://hal.science/hal-00437463>

Preprint submitted on 8 Dec 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

Jérémie Doucy^{1,2}, Habib Abdulrad¹, Patrick Giroux², Jean-Philippe Kotowicz¹

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 voyageur est le plus apte à définir un processus complexe de réservation de billets et d'hôtels. 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 prometteuse 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 fournit 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 paraît 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 apparaissent 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 définit 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 amène à la définition de chaînes non seulement génériques mais aussi dynamiques.

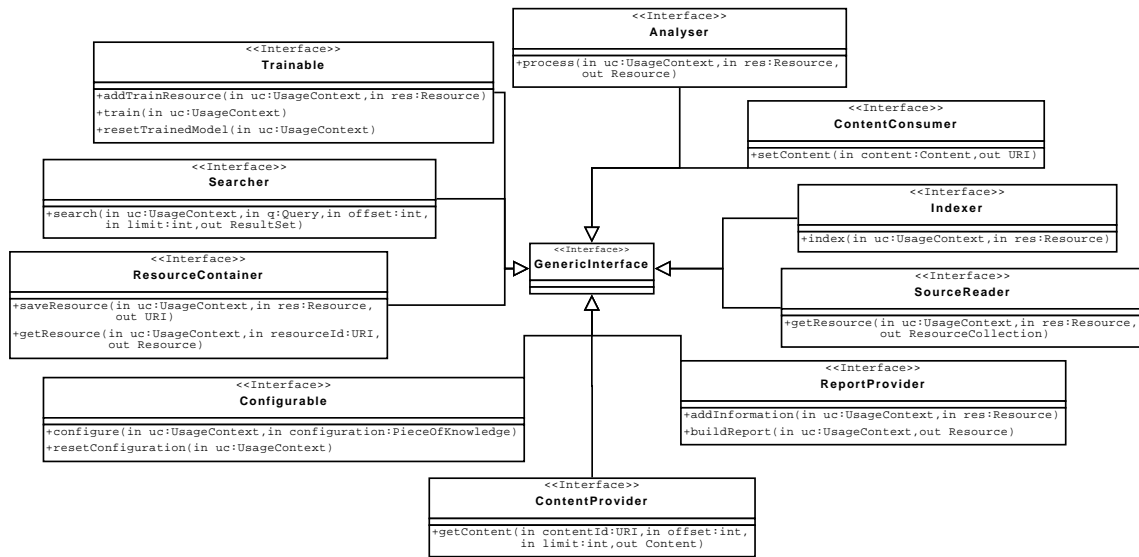


FIG. 1 – Interfaces g n riques d finies par la plateforme WebLab.

5. Application   la fouille de documents multim dia

Nous avons donc appliqu  ces nouveaux principes   la cr ation d’applications pour la fouille de documents multim dia. Actuellement, deux plateformes existent dans ce domaine UIMA [6] et WebLab [7]. Nous avons naturellement choisi le WebLab car cette plateforme est d velopp e en collaboration avec notre laboratoire. Ce choix est aussi motiv  par le fait que UIMA ne s’appuie pas sur les standards reconnus et semble moins souple pour la construction de cha nes. En effet, les  quivalents de patrons de cha nes, pour UIMA  tant pr d finis et difficilement modifiables.

Le WebLab est bas  sur le couple WSDL / BPEL en ce qui concerne la d finition des interfaces g n riques tout comme la cr ation de cha nes de traitements. Cette plateforme est donc coh rente avec nos travaux. Le WebLab d finit un ensemble de services g n riques, explicit s dans la figure 1, issue de l’expertise m tier d velopp e par notre laboratoire et ses partenaires.

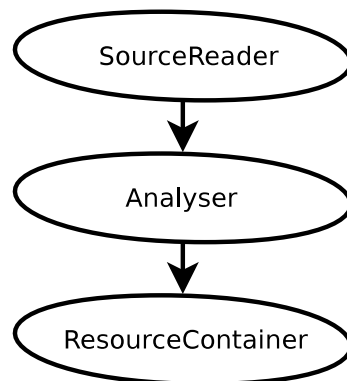


FIG. 2 – Exemple de « patron » de cha ne de traitements multim dia WebLab.

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.

<i>Interface</i>	<i>Service-name</i>	<i>Endpoint</i>
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 SOAP³ 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 désambigüiser chaque service du processus.

BPEL définit 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   l'annuaire un des *endpoints* impl mentant l'interface d finie dans le *partnerLink*. Imaginons un *partnerLink* li    l'interface *Analyser* et une op ration *invoke* qui est ex cut e sur ce *partnerLink*, le moteur choisira donc un des services disponibles impl mentant cette interface et l'appellera. Si l'on reprend l'exemple pr c dent, le moteur appellera un des quatre services impl mentant cette interface sans que l'on puisse choisir lequel. Cependant, un *partnerLink* est une variable en BPEL et peut donc  tre affect . Ceci permet de pr ciser   l'ex cution l'*endpoint* et le *service-name* que l'on veut invoquer. Si seulement le *service-name* est pr cis , le routeur du bus de services choisira un des *endpoints* d ploy s en utilisant ce *service-name* comme expliqu  pr c demment.

Cette d sambiguation dynamique permet d'instancier   la vol e des cha nes de traitements g n riques pr d finies. De plus, au cours de notre  tude des processus de traitements multim dia existants, nous avons mis en avant la nature lin aire de la majorit  des cha nes d'analyses. En effet le principe m me de la plateforme WebLab, et plus particuli rement de ses cha nes, est de permettre la succession de services d'analyses compl mentaires ou concurrentes dans le but d'enrichir les documents trait s. Cette suite est instanci e en BPEL par une succession statique d'appels   un *partnerLink* d clar  comme *Analyser*. Cette approche est satisfaisante dans un premier temps mais ne r pond pas compl tement au besoin de simplification de construction de processus BPEL. En effet, la mise   jour des cha nes lin aires est certes simplifi e,  tant donn e que la description de ce type de processus est r p titif et cyclique, mais n cessite toujours une certaine ma trise de BPEL. Nous avons donc envisag  une extension de cette solution comme explicit  dans la figure 3.

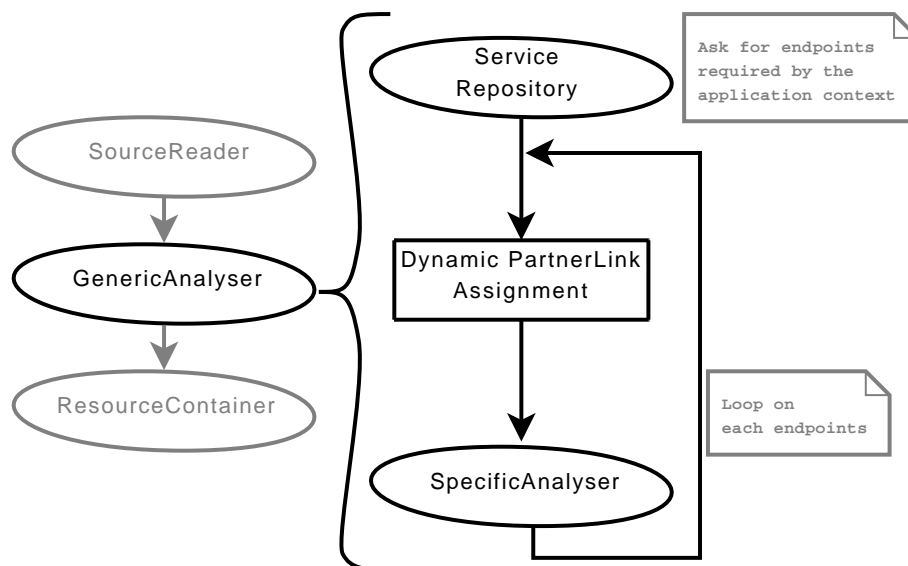


FIG. 3 – Dynamisation de la partie lin aire d'une cha ne de traitement.

Cette solution se base toujours sur le m me « patron » de cha ne mais se concentre sur la dynamisation de la partie analyse du processus. Cette partie  tant lin aire dans le cadre de ce « patron », la seule donn e n cessaire   l'instanciation de cette cha ne est la liste ordonn e des analyseurs   utiliser. Pour ce faire on boucle sur la liste d'*endpoints* retourn s par un annuaire de service. Ensuite pour chaque analyseur r cup r , on assigne dynamiquement le *partnerLink* avant d'appeler le service que l'on vient d'identifier   l'ex cution. Enfin, l'interface d'analyse prenant en param tre et retournant une ressource WebLab, on copie la r ponse du service dans la requ te 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'opérateur 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 précisément 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

  l'aide du raisonneur Alloy [8] et les premiers r sultats sont prometteurs.

Bibliographie

1. Charlton Barreto, Vaughn Bullard, Thomas Erl, John Evdemon, Diane Jordan, Khanderao Kand, Dieter K nigand 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. Hatley, 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, J r mie Doucy, G rard 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.