Java Complex Web Service Language

(Un langage de composition de services web complexes)

COULIBALY Demba

Université de Bamako
Bamako
Mali
demcoul@univ-bamako.edu.ml

RÉSUMÉ. Les propriétés d'interopérabilité et de faible couplage des services web lors de la mise en place des systèmes d'information hétérogènes, font d'eux actuellement l'une des meilleures solutions techniques d'implémentation des systèmes d'information. La mise en place facile des services web devient alors une nécessité. Vu les moyens actuels de cette mise en place et les difficultés liées, nous avons proposé une contribution avec un langage basé sur les langages BPEL (*Business Process Execution Language for Web Services*) et Java. Notre langage vient en complément au langage BPEL en proposant une méthode de composition qui présente un service web complexe comme une classe Java. Ce langage utilisera la flexibilité, l'expressivité et la facilité d'apprentissage de Java. Il s'appuie sur la réécriture des constructeurs de BPEL pour l'orchestration des services web. Notre langage utilise son propre environnement graphique pour la composition et propose une méthodologie UML (Unified Modeling Language) de conception des services web complexes.

ABSTRACT. The web services properties of interoperability and loose coupling during the implementation of heterogeneous information systems, now make them one of the best technical solutions for information systems implementation. The easy implementation of web services becomes a necessity. Given the current capacity of this development and the associated difficulties, we proposed a contribution with a language based on BPEL (*Business Process Execution Language for Web Services*) and Java. Our language is a complement to BPEL by proposing a method of composition that has a complex web service as a Java class. This language uses the flexibility, expressiveness and ease of learning of Java. It is based on rewriting BPEL constructors for web services orchestration. Our language uses its own graphical environment for composition and proposes its own UML (*Unified Modeling Language*) methodology to design complex web services.

MOTS-CLÉS : services web, Langage des services web, BPEL, Java, JCWSL, GEF, UML

KEYWORDS : web services, web services Language, BPEL, Java, JCWSL, GEF, UML

1. Etat de l'art des langages de composition des services web complexes

La composition des services web est une activité permettant la mise en œuvre de services complexes sur Internet. Elle est réalisée à l'aide de technologies, langages, et environnements. Dans cette section, nous présentons les technologies et langages de composition des services web ainsi que les plates-formes de composition de services web. La technologie principale des services web est le format XML.
XML [2] est un langage de balisage extensible qui a été mis au point par le XML Working Group sous l'égide du World Wide Web Consortium (W3C) en 1996. Les spécifications XML 1.0 sont reconnues comme recommandations par le W3C, ce qui en fait un standard. XML sert de base pour créer des langages balisés spécialisés : c'est un « *méta-langage* ». XML permet de représenter des données structurées ou des objets dans un fichier texte plat. Ceci est un exemple de fichier XML valide :

```
<Personne>
     <Nom>COULIBALY</Nom>
     <Prenom>Demba</Prenom>
</Personne>
```

XML permet aux utilisateurs de définir leurs propres structures de données. Ces structures sont appelées *méta-données*.

1.1. Les services web

Le terme « web services » regroupe un ensemble de technologies basées sur XML, permettant de créer des composants logiciels distribués, de décrire leurs interfaces et de les utiliser indépendamment du langage d'implementation choisi et de la plate-forme d'hébergement.
« *A Web Service is a software system designed to support interoperable machine to machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.* » [6]
SOAP[1], WSDL[2], UDDI[3] sont les technologies standard qui rendent possibles la construction et la publication de tels services.
L'infrastructure de base des services web présentée précédemment (WSDL, SOAP et UDDI), est suffisante pour implementer des interactions simples entre un client et un service web. Mais, si l'implémentation d'une logique métier implique l'invocation d'autres services web, il est nécessaire de combiner les fonctionnalités de plusieurs services dans un ordre bien défini. Dans ce cas, nous parlons d'un service web composé ou de composition de services web. Les services composés sont définis récursivement comme étant l'agrégation de services élémentaires et composés. En composant des services web, la logique métier du client est implémentée par plusieurs services. Ceci est analogue aux WFMS [7] (*WorkFlow Managment System*) où la logique applicative est réalisée en composant des applications autonomes.

1. *SOAP : Simple Object Access Protocol*
2. *WSDL : Web Service Description Language*
3. *UDDI : Universal Description, Discovery and Integration*

Les langages utilisés ont en commun les standards XML comme format des données et SOAP comme moyen de communication. L'utilisation de ces deux standards de la technologie des services web est une qualité qui fait d'eux tous portables, donc indépendants des systèmes d'exploitation.
Dans la plupart des cas, ils sont tous d'apprentissage difficile, car bâtis sur des concepts de balises.
Le tableau 1 représente un résumé des propriétés des langages existants par rapport à nos objectifs c'est à dire : *l'apprentissage*, *l'expressivité*, *l'extensibilité*, *l'exploitation*, *le développement d'un client correct*, *l'utilisation d'un environnement de conception*.
Les langages choisis ont pris un certain ascendant sur les autres langages du domaines. Il s'agit de : WSDL, ebXML (*Electronic Business using eXtensible Markup Language*), XLANG (*XML Business Process Language*), WSFL (*Web Services Flow Language*), BPEL, et BPELJ.
Ainsi, le langage WSDL est destiné à la description des services web. Ce qui le predestine aux services web simples sans pouvoir définir la logique des composants des services web.
Le langage ebXML met en contact des partenaires d'affaires pour échange. Son objectif était de créer un marché électronique unique et global.
Le langage XLANG réalise de l'orchestration de services web existants. Il est une extension de WSDL. Il fournit en même temps un modèle pour une orchestration des services et des contrats de collaboration entre services.
Le langage WSFL réalise la chorégraphie des services. Il est aussi basé sur WSDL pour la description de l'interface des services. C'est une spécification basée sur XML pour la description de compositions de services web en tant qu'élément d'une définition de processus métier.
Le langage BPEL peut réaliser des services web complexes par l'orchestration ou par chorégraphie, car il est basé sur XLANG et WSFL. Les difficultés liées à son apprentissage ont abouti à la mise en place d'environnements facilitant la réalisation de services web BPEL. Parmi ces environnement, nous pouvons citer les plus utilisés et connus (Oracle BPEL et Active BPEL).
Le BPELJ est venu améliorer les insuffisances de BPEL en terme d'expressivité et d'extensibilité. Avec l'incorporation de Java dans ce langage, il a obtenu un niveau d'expressivité et d'extensibilité plus élevé que BPEL, car il devient ainsi possible de définir ses propres classes Java par exemple pour exprimer certaines formes de données non prévues dans la sémantique de BPEL natif.
Dans le souci de mise en place de moyens plus simples et ne nécessitant pas d'environnement spécifiques pour leur mise en œuvre, nous proposons dans ce travail de recherche un langage beaucoup plus simple et plus expressif. Il sera basé sur BPEL et Java, mais à la différence de BPELJ, il ne sera pas un langage de balises, mais un script Java.

Critères / Langage	Types	Apprentissage	Expressivité	Extensibilité	Exploitation	développer un client	environnement de conception
WSDL	description	basé sur XML	création de ses propres types	non extensible	Serveur Web	facile	pas d'environnement dédié
ebXML	chorégraphie	plutôt facile	problème de représentation de données	non extensible	Serveur Web	facile	pas d'environnement dédié
XLANG	orchestration	basé sur XML	non expressif	non extensible	Serveur BizTalk	difficulté moyenne	pas d'environnement dédié
WSFL	chorégraphie	basé sur XML	non expressif	non extensible	Serveur IBM Websphere	difficulté moyenne	pas d'environnement dédié
BPEL	chorégraphie orchestration	difficile	peu expressif	non extensible	Serveur BPEL	difficulté moyenne	Active BPEL Oracle BPEL
BPELJ	orchestration orchestration	difficile	plus expressif que BPEL	extensible par Java Snippets	Serveur BPEL	difficulté moyenne	Active BPEL Oracle BPEL utilise Java

Tableau 1. *Tableau comparatif des langages existants par rapport à nos objectifs*

2. Le langage JCWSL

Comme constaté dans la section précédente, qu'il existe beaucoup de langages dans le domaine de la composition des services web. Il est à remarquer que la presque totalité de ces langages ont en commun la portabilité due chez certains à l'utilisation de XML et chez d'autres à l'utilisation des JVM (*Java Virtual Machine*) ou correspondants. Quand à la simplicité, elle varie selon le langage choisi, car certains sont simples par l'utilisation des environnements graphiques et d'autres tutoriels graphiques.
Notre travail concerne le développement d'un langage de programmation permettant d'obtenir l'abstraction du comportement observable d'un service web complexe et de générer un code exécutable en Java.
Le langage à mettre en place combinera BPEL (*pas assez expressif*) et Java (*pas assez fonctionnellement structurant*) BPEL et Java sont basés sur des paradigmes différents, car BPEL a une approche Workflow, tandis que Java a une approche Objet. JCWSL sera plus proche de Java que BPEL en terme de paradigmes.
JCWSL est un langage de script de Java enrichi des constructeurs de BPEL. L'objectif principal est de concevoir les services web complexes en utilisant le langage Java et définir en même temps le comportement du service web avec les constructeurs de BPEL.
JCWSL présente les avantages suivants :

– **niveau élevé d'expressivité** : Il supporte le langage Java, et donc offre un niveau élevé d'expressivité pour définir et concevoir un service web complexe,

– **transparence** : L'orchestration des services web locaux ou distants est intégrée dans le langage de façon transparente,

– **flexibilité** : Il offre deux types d'invocation de l'opération : invocation visible ou invisible. Une invocation visible apparaît dans le comportement de service, alors que l'invisible exécute l'invocation sans apparaître dans le comportement de service,

– **fortement couplé** : JCWSL prolonge d'une manière élégante et normale le langage Java. Ainsi, la manipulation des opérations et des messages sont très faciles et sont visibles et accessibles dans les parties implementation et comportement.

2.1. Concepts

Un service web complexe décrit dans le langage JCWSL est composé de deux blocs principaux qui sont le bloc des importations (`ImportBloc`) et le bloc de définition (`DefBloc`), comme sur la figure 1.

2.1.1. Les importations

C'est une sorte de préprocesseur permettant d'inclure dans le service web complexe JCWSL, l'ensemble des API Java et des services web basiques nécessaires. Nous prévoyons deux types d'importation qui sont les importations d'API Java (`import`) et les importations des services web à agréger dans notre service web complexe, représenté par `importBWS` (***import Basic Web Services***).
Les `importBWS` définissent dans JCWSL les liens avec les services basiques qui interagissent avec le service web complexe. Les commandes `importBWS` jouent en quelque sorte le rôle des PartnerLinks dans BPEL. Chaque service JCWSL doit avoir au moins une commande `importBWS` parce qu'un service complexe utilise au moins un service ba-

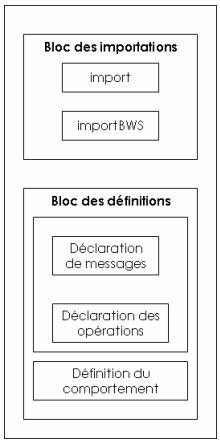

Figure 1. *Structure de JCWSL en blocs*

sique dans sa composition. Les `importBWS` permettent au service web complexe d'être le coordonnateur des interactions avec d'autres services, en faisant de l'orchestration.
Pour importer un service web existant, son adresse doit être connue au préalable.
Les importations de services web existants se traduisent en substance par la création d'un package Java pour chaque service importé. Il est crée pour chaque service importé un ensemble de classes Java constituant son Stub. Ainsi à l'aide de ce Stub, il devient facile d'invoquer le service par appel simple de ses méthodes comme celles d'une classe ordinaire Java.
Lors du déploiement du service web complexe, les packages des services importés doivent être archivés, car ces packages seront importés dans le code Java obtenu après la compilation du service complexe par le langage JCWSL.

Les `import` permettent l'importation des API Java nécessaires au service complexe. Ce sont des imports ordinaires de Java.

2.1.2. La définition du service web complexe

Ce concept représente le corps du service, une sorte de classe JCWSL faite à l'image d'une classe Java. Tout comme le bloc des importations, ce bloc est aussi composé de deux parties : le bloc des déclarations des messages locaux et des opérations locales et le bloc du comportement du service web complexe.
Le bloc des déclarations : Ce bloc permet de déclarer les messages et opérations du service web complexe. Les messages sont dits *messages locaux* et les opérations *opérations locales*.
Les messages représentent les données échangées entre le service complexe et les services basiques importés.
Les opérations quant à elles, sont définies pour les fonctionalités du service. Elles sont pour la plupart destinées à être utilisées dans les constructeurs du bloc comportement pour les interactions. Elles définissent l'interface du service web complexe ainsi que la description WSDL du service web complexe.
Le bloc comportement : Le comportement contient les interactions entre d'une part le service web complexe et d'autre part les services basiques importés. Ces interactions sont organisées en orchestration, car seul le service complexe connaît les services qui entrent dans le système. Le comportement contient des constructeurs WS-BPEL et du code Java.

L'ajout du code Java permet l'utilisation des instructions d'affectation, de boucles `for` et `do`, de l'instruction alternative `if then else`.

2.1.3. Les constructeurs

Nous présentons ici les principaux concepts sous forme de BNF. La BNF principale représentant le langage est la suivante :

```
CWSLanguage ::≡ <ImportBloc><DefBloc>
 ImportBloc ::≡ ("importBWS" <ID>=<STRING> ";" | "import" <ID>(.<ID>)* ";")+

 DefBloc ::≡ "public CWSDefinition" <ID> {[<declaration>]<main>}

 declaration ::≡ (<MessDeclaration>|<OpDeclaration>)*

 MessDeclaration ::≡ "public DefMessage" <MessageName>

                  "{"(<Variabletype> <VariableName>;)+"}"
 OpDeclaration ::≡ "public DefOperation" <OperationName>

            "(" <Input>":"<messagetype><ID> ")"
            | "("<Output>":"<messagetype><ID> ")"
            | "("<Input/Output>":"<messagetype><ID>,<messagetype><ID> ")"
            "{" <OperationBody> "}"
 OperationBody ::≡ (<java>|<execute>)*

 java ::≡ "JavaCode" "{" (instructionJava)+"}"
 main ::≡ "void main" "{"(<variables>|<process>|<Java>)*"}"

 process ::≡ sequence|receive|reply|invoke|throw|wait|pick|switch|while|

              flow|scope|compensate|<java>
```

2.2. Environnement de conception et de développement

Notre environnement de conception et de développement est nommée CWSE (*Complex Web Service Environnement*). C'est une plate-forme de type client riche Java développée à l'aide de l'environnement de développement intégré Eclipse.
L'environnement CWSE permet de concevoir un service web complexe JCWSL. Elle offre plusieurs modules permettant de définir, compiler et déployer un service complexe. Elle est composée de deux parties principales : une interface graphique de composition et un compilateur. Elle permet la composition statique des services web complexes. La plate-forme CWSE est un plug-in d'Eclipse.
L'interface graphique permet de concevoir le service sous forme de graphe avec des boutons conçus à cet effet.Le compilateur prend en entrée la description du service complexe écrit en JCWSL et génère le code exécutable correspondant en Java ainsi que le comportement en BPEL. Le compilateur est composé d'un analyseur et d'un générateur.

– L'analyseur assure l'analyse syntaxique et l'analyse sémantique du service web complexe.

– Le générateur, une fois les vérifications syntaxiques et sémantiques effectuées, produit automatiquement le code Java du service et son comportement observable abstrait.

2.3. Méthodologie UML pour JCWSL

Pour l'étude des services web en JCWSL, la méthodologie UML à suivre utilisera le processus unifié. Pour représenter les itérations, nous découpons la procédure en quatre itérations : l'expression des besoins, l'analyse conceptuelle, l'implémentation et le déploiement. A chaque itération, il est fortement conseillé de préciser davantage les besoins (objectifs) du service web à mettre en place.
La première itération consiste à prendre contact avec le sujet, cerner les besoins afin de définir les fonctionnalités premières du service web.
Pour la réalisation de l'analyse conceptuelle (réaliser la deuxième itération), nous avons choisi quatre diagrammes UML : les cas d'utilisation, les diagrammes des classes, le diagramme d'activités et le diagramme de séquences.
En se basant sur le MDA (Model Driven Architecture), des stéréotypes ont été créés pour représenter les différents concepts du langage JCWSL.

3. Bibliographie

[1] D. COULIBALY, « Un langage et un environnement de conception et de développement de services web complexes », *thèse de Doctorat, Université Paris-Dauphine*, juin 2009.

[2] S. NICOLAS,« Conception et mise en oeuvre d'une plate-forme pour la sûreté de fonctionnement des Services Web », *Institut National Polytechnique de Toulouse*,2006.

[3] T. BRAY, JEAN PAOLI, C. M. SPERBERG-MCQUEEN, EVE MALER, FRANÇOIS YERGEAU, « Extensible Markup Language (XML) 1.0 (Fourth Edition) », *W3C Recommendation 16 August 2006, edited in place 29 September 2006*, vol. 24, n° 6, 2006.

[4] S. HADDAD, P. MOREAUX, S. RAMPACEK, « Client synthesis for web services by way of a timed semantics », *In Proceedings of the 8th Int. Conf. on Enterprise Information Systems (ICEIS06)*, 2006.

[5] S. HADDAD, T. MELLITI, P. MOREAUX, S. RAMPACEK, « Modelling web services interoperability », *Proceedings of the 6th International Conference on Enterprise Information Systems, Porto, Portugal*, n° 4, 2004.

[6] D. BOOTH, H. HAAS, F. MCCABE, ERIC NEWCOMER, MICHAEL CHAMPION, CHRIS FERRIS, DAVID ORCHARD, « Web Services Architecture W3C Working Group Note 11 », 2004.

[7] W. VAN DER AALST, K. VAN HEE, « Workflow Management, Models, Methods, and Systems », *The MIT Press, London*, 2002.

[8] M. DUMAS, M.-C. FAUVET,« Intergiciel et Construction d'Applications Réparties », *Licence Creative Commons (http : $//creativecommons.org/licenses/by-nc-nd/2.0/fr/deed.fr$)*, 2006.

[9] D. ALMAER, « Creating Web Services with Apache Axis », *ONJava.com (http : $//www.onjava.com/lpt/a/1578$)*, 2002.

[10] CEN/ISSS, W/S EBES, « Technologie ebXML », *(http : $//www.ebxml.eu.org/French/TechnologieebXML.htm$)*, 1999-2009.

[11] R. SYLVAIN, « Sémantique, interactions et langages de description des services web complexes », *Université de Reims Champagne-Ardenne*, 2006.