Implémentation à l'aide de BPEL de trois processus d'agrégation de composants, dirigée par les modèles

Anis Masmoudi*, **— Gilbert Paquette *— Roger Champagne **

**LICEF : Laboratoire Informatique Cognitive et Environnements de Formation, Montréal, Québec, Canada.*
*** ÉTS : École de technologie supérieure, Montréal, Québec, Canada.*
{anis.masmoudi, gilbert.paquette}@licef.teluq.uqam.ca,
roger.champagne@etsmtl.ca

RÉSUMÉ. *Plusieurs organisations qui œuvrent dans le domaine d'apprentissage à distance utilisent le composant logiciel comme unité de base pour construire leur système. Ils ne développent presque plus de nouveaux composants, mais ils les réutilisent et appliquent des réingénieries pour des fins d'adaptation aux nouveaux contextes. Ceci prouve que le développement logiciel par agrégation des composants est un sujet d'intérêt. Cette branche du génie logiciel constitue un des axes fondamentaux du projet canadien LORNET (Learning Object Repositories' NETworsk). Cet article donne suite à des travaux publiés l'an dernier, proposant principalement d'adjoindre aux composants logiciels certains types de métadonnées que nous avons intitulé SOCOM (SOftware COmponent Metadata). Nous avons défini trois types d'agrégations avec des exemples concrets. Dans le présent article, nous rappelons brièvement ces métadonnées et les catégories d'agrégation existantes et proposées et nous utilisons un langage d'exécution de processus métier intitulé BPEL (Business Process Execution Language) pour implémenter des catégories d'agrégation tels que : Collection, Coordination et Fusion.*

ABSTRACT. *Many organizations research on develop eLearning environments based on software components as their system's base construct. They don't develop new components, but they reuse existing ones. They apply software engineering concepts such as reengineering, reverse engineering and software components reuse. System development based on software components is an important issue also in LORNET project (Learning Object Repositories' NETworks). This paper extends some previous work. We remind briefly our SOCOM (SOftware COmponent Metadata, metadata structure that characterizes software components) and we explain shortly our aggregation classification based on three attributes from SOCOM. Afterwards we use BPEL as a business process execution language to help us to implement our three designed aggregation's categories: aggregation by collection, by coordination and by fusion.*

MOT-CLÉS : *Composant, Agrégation, Développement dirigé par les modèles, Développement basé sur les composants, Modélisation, Métadonnées de composants logiciels, BPEL.*

KE YWORDS: *Component, Aggregation, Model driven development, Component based development, Modeling, Software component metadata, BPEL.*

Catégorie de la soumission : Chercheur

1. Introduction

Les environnements d'apprentissage à distance ne cessent de se multiplier en utilisant plusieurs technologies de développements. Ils offrent différents services à leurs apprenants, leurs concepteurs, leurs tuteurs et leurs administrateurs. Dans ce même contexte s'inscrit l'initiative du projet LORNET. Ce dernier vise plus loin qu'une simple application d'apprentissage. Un de ses objectifs principaux est la mise en place d'un système générateur de systèmes d'apprentissage. Ces systèmes à leur tour génèreront des applications spécifiques conformément à l'une des disciplines scientifiques. Ces applications sont accessibles par les tuteurs (chargés de cours) pour assister les apprenants lors du suivi d'un cours à distance. Chaque apprenant utilise des ressources pédagogiques et en fournit d'autres. Cette cascade, supportée par le système TELOS (*Tele-Learning Operating System*) du projet LORNET, est conçue par des agrégations de composants logiciels et de ressources pédagogiques existantes.

L'agrégation de composants logiciels dirigée par les modèles montre plusieurs problèmes dans la communauté du génie logiciel. Bien qu'on réussi souvent à modéliser les composants par des modèles UML, les problèmes de connectivité et de conformité du code par rapport au modèle demeurent (Selic, 2003). Cet article ne résout pas globalement le problème, cependant nous montrons à travers des illustrations qu'un langage d'exécution de processus métier peut modéliser une partie des besoins des applications pédagogiques. Ce modèle sera connecté aux services des composants au moment de la conception. Ensuite, ce même modèle sera exécuter et suivi lors de son exécution.

Notre but à long terme est la conception d'une méthodologie qui assiste un ingénieur agrégateur dans la réalisation d'une agrégation de composants logiciels dirigée par les modèles. Le travail décrit ici prépare ces ingrédients, qui aideront à la mise en pratique de notre méthodologie d'agrégation, mais ne le décrit pas en détail. Ce dernier fera l'objet des travaux futurs dans le cadre du projet LORNET.

Dans ce qui suit, nous commencerons par rappeler, dans la section 2. , le choix des métadonnées des composants logiciels. La section 3. rappelle différentes formes et catégories d'agrégations retenues à partir de la littérature. Puis nous présenterons les trois catégories d'agrégations adoptées en se basant sur trois des métadonnées proposées. Nous justifions notre choix de la technologie BPEL et nous exposons brièvement la problématique courante du modèle et du code d'un composant dans la section 4. Techniquement, notre modélisation d'agrégation des composants logiciels utilisait le formalisme UML[1] en se basant sur son modèle de composants. Pour mieux comprendre la structure de notre nouveau modèle BPEL des composants, nous tenons à présenter une correspondance entre le modèle de composant UML2 et son équivalent en BPEL[2]. Ceci sera fait dans la sous section 4.3. Cette corrélation sera utile pour expliquer comment un processus BPEL

[1] UML (Unified Modeling Language) : http://www.uml.org/
[2] BPEL (Business Process Execution Language): http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

consomme les services externes et comment il en fournit d'autres. Ceci nous facilitera la tâche pour détailler, dans la section 5, trois exemples d'agrégations utilisant le langage BPEL. Pour chaque exemple, nous rappelons la structure du processus BPEL et les spécificités du langage utilisées. Pour terminer, nous rappelons l'objectif de cet article et les différents choix adoptés pour servir le paradigme d'agrégation de composants. Aussi, nous décrivons brièvement notre méthodologie qui va réutiliser les concepts et les outils que nous avons développés à ce propos afin d'assurer des agrégations de composants de bout en bout.

2. Caractérisation des composants logiciels par des métadonnées

Nous avons développé antérieurement (Masmoudi et al., 2005) une structure de métadonnées des composants logiciels. Cette structure couvre trois vues du composant logiciel et identifie des attributs techniques, des attributs non fonctionnels et des attributs métiers. Par cette caractérisation, nous avons voulu rendre la description d'un composant logiciel accessible par différents types d'intervenants dans le processus de développement logiciel à savoir analyste, concepteur et programmeur.

Les attributs techniques décrivent le comportement du composant interne et externe. En effet, d'une part, ils décrivent les services offerts par les composants pour servir son environnement externe, soit le contexte où il est développé. D'autre part, ces attributs décrivent les services requis par le composant pour assurer son fonctionnement interne et délivrer ses produits. D'autres attributs du composant sont couverts par notre structure comme l'identificateur local et global du composant, le nom du composant, l'endroit de son paquetage technique, le degré d'accessibilité, le niveau de couplage, la couche logicielle à laquelle il appartient, son modèle de composant UML et la catégorie de l'agrégation du composant.

Les attributs non fonctionnels sont des aspects dits non techniques (aussi nommés attributs de qualité dans la communauté d'architecture logicielle) du composant mais qui ont un impact sur une réutilisation potentielle de ce composant dans un domaine autre que celui de sa création. Nous parlons des aspects tels la fiabilité, la sécurité, l'extensibilité, le ou les droits d'utilisation, la performance, l'intégrité, etc. Ces attributs couvrent aussi certains aspects de l'environnement du composant au niveau des plates-formes de développement, de déploiement, de configuration, de la persistance (données) et des langages de programmation (OMG, 2003).

Les attributs métiers, aussi nommés attributs du domaine par plusieurs, visent à capturer des informations sur le contexte du composant. Dans notre cas, nous décrivons les scénarios métiers dans lesquels participe le composant, qui ne sont autres que des scénarios pédagogiques. Aussi, ces attributs sauvegardent les métadonnées pédagogiques ayant le format du LOM[3] dans notre contexte.

[3] LOM (Learning Object Metadata) : métadonnée des objets d'apprentissage.

3. Catégorisation de l’agrégation des composants

Une revue de la littérature sur les termes les plus utilisés pour définir la mise en relation d’association de deux composants nous montre que plusieurs auteurs utilisent association, composition, agrégation, intégration, connexion et assemblage pour décrire des relations de structure et de logique différentes entre deux composants logiciels. Dans ce qui suit, nous expliquons les termes les plus utilisés puis nous présentons brièvement nos catégories d’agrégation qui regroupent la majorité des termes trouvés dans la littérature.

3.1. Les différents termes d’association des composants logiciels

Une association de composants est une mise en relation de deux composants. Ces derniers communiquent moyennant une couche de services fournis et requis. Chaque composant appartient à un domaine spécifique et possède une logique métier qui lui est propre. Il s’agit d’une simple connexion de service entre composants. De ce fait, une agrégation de composants est une association complexe mettant en relation de composition deux composants selon une logique métier et un contexte spécifique (Ivers et al., 2004). On entend par composition, une manière de structurer et de construire des applications à partir de composants logiciels. Villabos se basent sur trois types de composants – boite noire, boite blanche et boite grise – pour définir les trois types de composition (Villabos, 2003) :

- Composition boite noire : il s’agit d’une intégration du monde d’exécution des composants. On ne connaît pas de composition fonctionnelle de l’ensemble. La composition des besoins non fonctionnels se fait à travers des langages de haut niveau d’abstraction. Techniquement, la composition se fait du côté des méthodes et non des instructions. On parle ainsi de connexion de composants.

- Composition boite grise : la composition ne se fait pas dans le monde de l’exécution. Les composants sont réutilisables du point de vue service. La composition se fait au niveau des instructions et non au niveau de la méthode. Il s’agit dans ce cas de coordination de composants.

- Composition boite blanche : dans la composition de niveau boite blanche, il n’y a pas davantage d’abstraction. L'inclusion des structures non fonctionnelles se fait directement sur les structures du programme. On appelle cette composition "intégration" la composition de niveau boîte blanche.

L’assemblage de composants est une approche similaire à celle vue dans l’industrie mécanique telle que l’assemblage de pièces automobile pour la construction d’autos. Cette approche utilise les trois types de composition cités ci-dessus pour la création d’application logicielle par assemblage de composants.

3.2. Les trois catégories d'agrégations proposées

La caractérisation des composants logiciels n'est pas suffisante pour savoir comment les agréger, d'où la nécessité de catégoriser leur agrégation en se basant sur des attributs clés. Nous avons proposé auparavant (Masmoudi et al., 2005) trois catégories d'agrégation de composant suite à une mise en relation ensembliste. Nous utilisons trois des attributs de métadonnées techniques présentés ci-dessous pour qualifier le type d'agrégation des composantes : collection, coordination ou fusion. Pour décider du type d'agrégation, nous considérons les dimensions suivantes :

- **CL** : couche logicielle à laquelle le composant appartient. On distingue les couches suivantes : Présentation (P), Traitement (T), Middleware (M), Données(D). On rappelle que la couche P englobe tout le code utilisé pour définir les éléments graphiques des interfaces usagers. Celle de traitement regroupe l'ensemble des classes qui manipulent les objets du système pour implémenter les besoins fonctionnels. La couche M définit des classes qui servent d'interface pour l'adaptation des requêtes entre les couches T et D. Nous trouvons dans cette couche un connecteur vers une base de données ou un engin de routage de requêtes. La couche D contient les entités qui gèrent les objets de données. Elle encapsule aussi toute la structure de la base de données de l'application ou du système en question.

- **DA** : degré d'accessibilité au code du composant selon sa structure interne. On distingue ici trois situations : composant boite noire (BN), composant boite blanche (BB), composant boite grise (BG).

- **PC** : potentiel de couplage du composant selon les relations du composant avec les composants de la même couche ou de couches différentes. On distingue ici le couplage horizontal (CH), le couplage vertical (CV) et le couplage mixte (CM), à la fois horizontal et vertical.

Pour une agrégation par collection, les composants de l'application A (CpA) communiqueront avec les composants de l'application B (CpB) uniquement au niveau des couches logicielles **P** et **D** .Ils présentent un potentiel de couplage horizontal à ce niveau. Ces composants possèdent un degré d'accessibilité faible ou moyen ; conséquemment, nous parlons dans ce cas de composants boites noires ou de composants boites grises.

Dans l'agrégation par coordination, les composants de l'application A (CpA) et ceux de l'application B (CpB) communiquent généralement suivant un couplage mixte, à la fois horizontal et vertical Les composants de l'agrégation entrent en relation sur une même couche ou des couches adjacentes verticalement. Les composants qui entrent en jeu sont de type boite grise ou boite blanche. Autrement dit, nous avons un accès minimal ou complet au code du composant. Du point de vue couche logicielle, les composants qui entrent en jeu appartiennent aux couches de traitement (T) ou de middleware (M).

Pour l'agrégation par fusion, les composants des applications appartiennent obligatoirement à la même couche logicielle et présentent un potentiel de couplage

horizontal fort. À la fin du processus, nous obtiendrons uniquement un seul composant physique. Pour ce faire, le degré d'accessibilité au composant doit être fort**.** Il s'agit des composants de type boite grise ou blanche nous donnant accès à leur code partiellement ou totalement. L'agrégation en question se réalise selon deux modèles : le premier fait intervenir un troisième composant fédérateur qui va englober les deux composants. Le second est le modèle tout-partie (i.e. un composant englobe l'autre).

Nous avons réalisé les trois types d'agrégation dans les projets EduSource [4] et LORNET [5] au sein du laboratoire LICEF de la Télé-université. Les exemples sont détaillés et disponibles dans (Masmoudi & al., 2005).

4. Pourquoi le langage BPEL ?

Pour développer un processus d'agrégation complet avec le langage BPEL, trois étapes sont nécessaires. Le premier consiste à formaliser les besoins d'agrégation dans le langage BPEL sous forme d'un fichier XML d'extension «.bpel » conforme à la structure BPEL. Ceci pourra être réalisé avec un outil de conception graphique ou textuel tel que «Oracle jDeveloppер Designer» (BPEL Oracle Group, 2005) ou « Eclipse BPEL » (Eclipse BPEL, 2006). Ensuite, il faut déployer ce fichier dans un serveur d'application tel que le serveur BPEL d'Oracle. Une fois exécuté, le processus a besoin d'un outil pour visualiser son avancement. Par exemple, Oracle propose une console qui permet de déclencher et de gérer une ou plusieurs instances d'un processus BPEL et d'assurer son suivi lors de l'exécution.

BPEL connaît un bon suivi et une adoption incrémentale par une large communauté internationale. Il est supporté par une trousse d'outils diversifiée et complète depuis la conception graphique simplifiée de ses processus à leur suivi au moment de l'exécution. Cette complétude en fait un bon candidat pour la mise en place de processus métier exécutable de bout en bout. En effet, d'autres langages proposent des modèles de composant à l'aide d'outils appropriés pour satisfaire des besoins d'analyse, de conception et d'implémentation. Cependant, les volets d'exécution et de suivi sont rarement supportés à cause du manque de connectivité et de conformité des propriétés des modèles aux codes des composants d'une part et de la complexité de mise en place un serveur d'exécution de ses modèles d'autre part. BPEL est l'un des langages d'agrégation de composants à l'étude dans le cadre du projet LORNET.

---

[4] Projet eduSource:http://edusource.licef.teluq.uquebec.ca/ese/fr/index.jsp
[5] LORNET: Learning Object Repository NETwork(www.lornet.org)

4.1. BPEL un langage d'exécution de processus métier

BPEL bénéficie de quinze années de travail sur les langages de processus XLANG[6] et WSFL[7]. Soumis à OASIS en mars 2003 (OASIS Group, 2005), le langage gagne le support de la majorité des vendeurs industriels. On le qualifie de langage intégrateur de services discrets dans un flux de processus métier de bout en bout. BPEL adopte la structure WSDL comme modèle des composants qu'il agrège. Il utilise le langage XML comme modèle de données. Les traitements sont implémentés par des échanges de messages structurés de type synchrones et asynchrones. Ces messages – associés aux activités – forment une séquence logique formant le processus qui à son tour peut être synchrone ou asynchrone. De plus, il implémente ses flux en tenant compte des différentes exceptions de traitement possibles sous forme hiérarchisée similaire aux autres langages de programmations (Java, C++, C#, etc.).

Les processus BPEL ont une durée de vie bien déterminée et leur cycle de vie peut être contrôlé par l'implémentation de contraintes spécifiques. On peut gérer plus d'une instance des processus BPEL à partir de la console de son serveur. BPEL est qualifié de langage de processus récursif. En effet, le processus agrège plusieurs services sous forme de services Web ou autres et il peut, à son tour, être considéré comme un service Web du moment qu'il dispose d'une description en WSDL. Par conséquent, il peut participer à d'autre processus d'agrégation.

Dans plusieurs réalisations techniques, les agrégations des composants logiciels utilisent essentiellement le modèle des composants UML ou UML 2. Dans ce qui suit, nous présentons brièvement la structure XML du processus BPEL (sous section 4.2). Ensuite, nous faisons une corrélation de cette même structure avec celle du modèle UML (sous section 4.3). Ceci facilite la compréhension et la vision du processus BPEL en tant que composant agrégateur.

4.2. Le langage BPEL : Structure et outils

La force du langage BPEL se manifeste dans la structure de son processus qui tient sur un fichier XML unique (Figure 1). Dans le premier bloc (B1) de ce fichier, nous trouvons les partenaires qui offrent les services au processus. Pour chaque partenaire, on indique le nom, le type de service qu'il offre et le rôle (fournisseur, consommateur). Le deuxième bloc (B2) présente toutes les variables de traitement. Pour chaque variable, on indique le nom et le type de message utilisé. Le troisième bloc (B3) liste l'ensemble des activités de traitement sous la forme d'une séquence logique d'appels et de retour d'appels aux composants partenaires. Il renferme aussi

[6] XLANG (XML business process LANGuage): http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm
[7] WSFL (Web Service Flow Language) : http://www-128.ibm.com/developerworks/webservices/library/ws-ref4/index.html?dwzone=webservices

la liste des échanges d'information entre les variables déclarées par le deuxième bloc. L'activité « invoke » réalise effectivement l'appel à un des services offerts par le partenaire. Elle est caractérisée par un nom, le nom du partenaire invoqué, le type de port utilisé et les variables en entrées et en sorties de l'appel. L'activité « assign » est souvent utilisée pour assigner à des variables le contenue d'autres sous la forme de message au sein du processus.

Figure 1. La structure principale du fichier XML d'un processus BPEL

Le fichier du processus BPEL est indépendant des plateformes BPEL, En effet, nous avons conçu un processus BPEL par l'outil « Eclipse BPEL » et nous l'avons déployé dans un serveur BPEL d'Oracle. Aussi, nous avons déployé un autre processus BPEL dans le serveur « ActiveBPEL »[8] qu'on a conçu avec l'outil « jDeveloper BPEL Designer ». Ceci montre un premier niveau de réutilisation des agrégats BPEL dans divers serveurs BPEL. Nous rappelons que ces processus BPEL possèdent des descriptions WSDL pouvant être consommées en services web dans d'autres contextes, d'où le deuxième niveau de réutilisation.

[8] ActiveBPEL : un serveur d'exécution BPEL, http://www.activebpel.org/

4.3. Du modèle des composants UML à BPEL

Le modèle de composant UML2 (Figure 2) considère le composant comme une unité de traitement qui consomme et fournie des services par des symboles spécifiques. De son coté, le modèle graphique BPEL utilise une cascades de rectangles pour présenter sa séquence d'activités qui consomme et fournie des services (BPEL Oracle Group, 2005). Plus précisément, l'activité « invoke » est utilisée pour appeler des services externes. Il reçoit les appels à ses services par l'activité « receive » et retourne les résultats du traitement interne à l'appelant par l'activité « reply ». Les services requis par le processus BPEL lui sont offerts sous la forme de « PartnerLink ». Ces partenaires BPEL correspondent aux composants A et B (voir Figure 2). Pour finir avec la corrélation, les échanges entres les deux composants (encadré en pointillé) sont équivalentes à la séquence d'activités du processus BPEL. Il est important de noter que *les appels entre composants UML, qui se font par des invocations de méthodes objets, trouvent leur correspondant dans le processus BPEL à travers les échanges de messages structurés de document XML*.

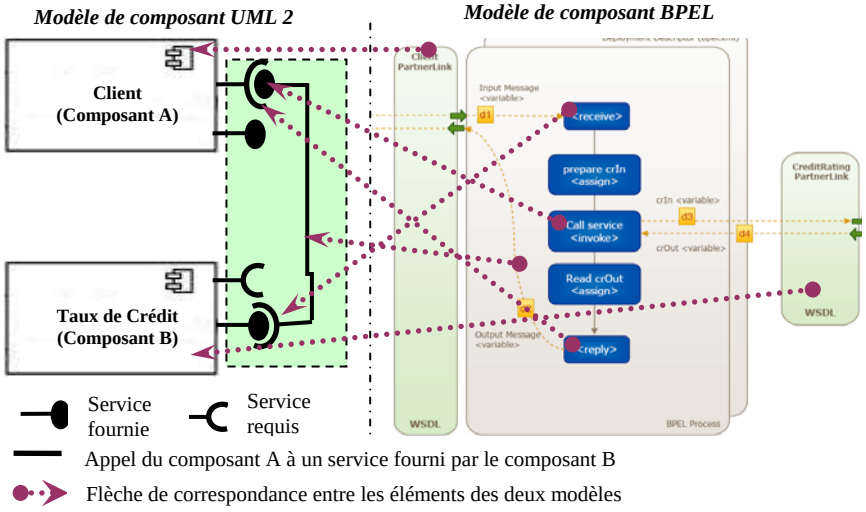

Figure 2. Les modèles de composant logiciel UML 2 et BPEL

5. Implémentation des trois types d'agrégations dans BPEL

Dans ce qui suit, nous verrons comment cette structure BPEL peut matérialiser les trois types d'agrégations détaillés dans la section 3.2 à savoir: collection, coordination et fusion.

5.1. Implémentation BPEL d'une agrégation par collection

L'agrégation par collection consiste à mettre en association des composants qui communiquent entre eux par la couche de présentation ou la couche de données. Nous avons utilisé la structure XML du processus BPEL pour assembler les appels aux services de ces composants. Nous avons réalisé l'exemple de collection de données de différentes sources. L'objectif était de fournir une fiche complète d'un étudiant dans un système d'apprentissage (informations personnelles et scolaires intégrées ensemble). À ces deux types d'informations sont associés respectivement deux composants de données *CompInfoA* et *CompInfoB*.

Techniquement, des activités « invoke » ont été utilisées pour extraire l'identificateur de l'étudiant « studentId » depuis un formulaire validé par le serveur d'application. Ensuite, cet identificateur a été utilisé pour invoquer un service Web depuis *CompInfoA* qui retourne la fiche personnelle correspondante. Cette fiche est transmise sous forme d'un document XML à un autre service Web fourni par *CompInfoB* qui retournera la fiche scolaire associée.

En ce qui concerne l'agrégation au niveau de la couche présentation, nous avons conçu deux partenaires « partnerLink » sous forme de deux composants de présentation (des interfaces usager) (*CompInterfA*, *CompInterfB*). *CompInterfA* fournit la description des objets d'apprentissages formatés sous la forme de spécifications IEEE LOM[9] en lui passant comme paramètre le lien URL du site web qui servira de support pour les étudiants lors du suivi du cours à distance. *CompInterfB* fournis le style d'affichage selon le patron de style demandé. Dans le processus BPEL, des activités « invoke » sont utilisées pour appeler ses composants et extraire leurs données sous un format LOM, puis les afficher selon un style spécifique. À la fin de l'exécution, le processus retourne une fiche ordonnée pour l'étudiant résumant l'ensemble des ressources pédagogiques nécessaires pour l'aider lors du suivi de son cours.

En se basant sur la définition d'agrégation par collection (Section 3.2), nous retrouvons que le processus BPEL a agrégé deux composants de données ou de présentation avec un couplage horizontal. Quant au degré d'accessibilité, *CompInfoA*, *CompInfoB*, *CompInterfA* et *CompInterfB* sont quatre composants boites grises parce qu'ils exposent leurs services sous forme de Web Service.

5.2. Implémentation BPEL d'une agrégation par coordination

La coordination dans un contexte pédagogique est une orchestration. Nous entendons par orchestration la mise en relation ordonnée d'activités, de ressources pédagogiques, d'acteur et de rôle de façon synchrones ou asynchrones pour servir

[9] LOM IEEE: IEEE Learning Object Metadata (http://www.imsglobal.org/metadata/mdv1p3pd/imsmd_bestv1p3pd.html)

un scénario d'apprentissage spécifique. Le langage BPEL propose des structures de contrôle assez élaborées pouvant être utilisées dans la conception de processus. L'orchestrateur ou bien le chef d'orchestre n'est autre que le processus en soi. Il coordonne les services des composants externes implémentés sous formes de « partnerLink » et des interventions d'acteurs via des interfaces usagers pour faire avancer le processus dans une logique séquentielle conforme au contexte d'apprentissage étudié. BPEL contrôle l'attente au sein du processus par l'activité « wait » qui peut être paramétrée par une durée temporelle ou bien déclenchée par une tierce personne utilisant des interfaces usager appropriées.

Dans ce qui suit, nous présentons un exemple d'agrégation par coordination intitulé « Révision d'un document dans un processus d'apprentissage ». Dans ce cas, le processus BPEL est utilisé pour assurer la révision en parallèle d'un document par quatre participants. Trois d'entre eux feront une revue technique, le quatrième est un décideur. Il revoit les commentaires des trois autres et décide de l'approbation finale ou non du document en question. On appellera dorénavant le processus étudié *révisionDoc*.

Pour établir la liste de séquence d'activités, *révisionDoc* a besoin des composants suivants: D'abord, un composant de présentation intitulé *client* initie le processus en publiant le document à réviser dans un serveur dédié utilisant le message sous forme de document XML de la Figure 3. Ensuite, un composant de traitement *contrôleur de tâches* est nécessaire pour gérer les tâches de révision et déclencher une alerte par courriel des trois réviseurs. Ce dernier prend les informations du client et crée quatre tâches de révision pour chaque étudiant. Il communiquera avec le composant de données appelé *gestionnaire d'usager* qui lui fournit les fiches personnelles des réviseurs. Avec ses fiches il utilise le *gestionnaire de courrier* pour envoyer des messages aux étudiants concernés. Ce contrôleur de tâches communique ses informations sur les tâches et les usagers au *gestionnaire de tâches* qui offrira le routage des tâches d'un réviseur à un autre de façon ordonnée. Ce dernier fournira des interfaces usagers pour permettre à chaque étudiant réviseur d'accéder à son compte, de consulter le document, d'ajouter des commentaires et de valider son contenu. Ce gestionnaire de tâches retournera au client de départ la décision finale sur le document lorsque tous les réviseurs auraient validé le document étudié. Le décideur sera averti par courriel quand les trois autres auront fini la révision. En ce moment, il doit réviser et valider à son tour le document et les annotations de ses collègues.

Pour le processus BPEL les composants sont des partenaires offrant des services via leur description WSDL. Techniquement, ce processus utilise les mêmes activités que celles décrites dans la section précédente. Pour réussir cette orchestration, des structures de contrôle élaborées sont utilisées telles que « switch », « while » et « If then, else ». Elles sont utiles pour l'extraction des fiches personnelles des étudiants, le routage des tâches et le changement de comportement du processus dépendamment de l'intervention des réviseurs. Aussi, nous avons implémenté l'aspect asynchrone par l'utilisation de l'activité « wait » qui est nécessaire afin que

le processus attende la fin de la tâche de révision des trois étudiants pour déclencher la tâche de révision du décideur.

En revenant à la définition d'une agrégation par coordination, on retrouve dans cet exemple un couplage mixte du composant de traitement du *gestionnaire de tâches*. Ce dernier couple horizontalement avec le *contrôleur de tâches* et le *gestionnaire de courrier* et verticalement avec le *client* (Composant de présentation) et le *gestionnaire d'usagers* (Composant de données). Ces composants offrent leurs fonctionnalités sous forme de service web ainsi ils sont qualifiés de composant boites grises.

```
<MyDocReviewerProcessRequest  xmlns="http://xmlns.oracle.com/MyDocReviewer">
   <documentTitle>Document review process</documentTitle>
   <documentName>Document review </documentName>
   <URI>file:C:demos\DocumentReview\DocProcess.pdf</URI>
   <assignee>amasmoudi</assignee>   <assignee>gpaquette</assignee>
   <assignee>rchampagne</assignee>  <reviewer>omarino</reviewer>
</MyDocReviewerProcessRequest>
```

Figure 3. Le message XML d'entrée pour déclencher révisionDoc

5.3. Imploémentation BPEL d' une agrégation par fusion

Dans note exemple l'agrégation par fusion utilise des composants de type boite grise et boite blanche. En effet, le langage BPEL propose plus qu'une agrégation des services web, il tolère des appels à des composants par leur code natif. Dans notre cas, nous avons utilisé BPEL pour faire des appels au code Java sous plusieurs formes (Lehmann, M, 2005). En effet, les processus BPEL sont capables de communiquer avec des entités EJB (*Enterprise Java Beans*), des composants Java via les connecteurs JCA (*Java Connector Adapter*), les services de messageries Java JMS et les adaptateurs des bases de données. Des langages de requêtes pour l'extraction de données des documents XML tels XPath[10] et XQuery[11] sont aussi supportés. De plus, on peut utiliser du code Java embarqué dans les balises BPEL moyennant l'activité « bpelx:exec ».

Le résultat d'une agrégation par fusion utilisant le langage BPEL est un paquetage physique unique qui englobe la structure XML du processus avec les appels natifs Java et le code des composants agrégés.

L'échange de variables se fait via les fichiers XML. Le code java lit les variables des fichiers XML puis les utilise pour réaliser des traitements spécifiques selon le

[10] XPath: http://xmlfr.org/w3c/TR/xpath/
[11] XQuery: http://www.w3.org/TR/xquery/

contexte. À la fin, ce même code remet à jour la variable ou la structure XML modifiée. On parle d'une agrégation fine au niveau des variables favorisant ainsi la fusion des traitements BPEL et Java dans un même fichier XML. La fusion ne suppose pas le fait que seul le processus BPEL appelle les composants Java, mais il prévoit aussi un appel dans le sens contraire. Cela est possible vu que chaque processus BPEL est décrit par un fichier WSDL. En fait, le composant Java utilise le cadre conceptuel WSIF[12] pour construire un client sous forme d'un service web. Ce client est capable de faire des appels au processus BPEL et d'utiliser ses variables internes via des services que le processus fourni.

L'exemple de la Figure 4, qui représente une partie du fichier BPEL de l'agrégat par fusion, montre une interaction bidirectionnelle entre le processus BPEL et une entité EJB *« StudentHome »*. Suite à un appel sous forme d'un service web à un composant de traitement du *gestionnaire d'usagers*, le processus BPEL reçoit via l'activité « receive » la fiche de l'étudiant. Il affecte cette fiche, qui a la forme de document XML, à une variable interne « Input ». Cette variable est récupérée par le code Java pour qu'elle soit sérialisée et manipulée en objet Java « *StudentApplication* ». Depuis cet objet, on extrait l'adresse de courriel d'un étudiant qu'on passe à une méthode de l'entité EJB pour récupérer l'identificateur de cet étudiant. Le but final de cet échange est d'obtenir l'identificateur de l'étudiant ayant son adresse de courriel et de retourner sa fiche complète au client qui a fait la demande. Autrement dit, le processus BPEL recevra en entrée une fiche incomplète de l'étudiant et retournera un document XML de sa fiche complète.

Nous retrouvons dans cet exemple la définition d'une agrégation par fusion. En effet, le processus BPEL de cet exemple est un composant fédérateur qui appelle via ses services web le *gestionnaire d'usagers*, un composant de traitement boite grise et *« StudentHome »* , un composant de traitement EJB de type boite blanche. Le couplage est bien horizontal au niveau la couche de traitement.

[12] WSIF (Web Service Invocation Framework) : Cadre conceptuelle pour l'invocation dynamique de service web.

```
<variables><!-- input of this process -->
   <variable name="input"
               messageType="tns:StudentFlowPlusRequestMessage"/
   <variable name="StudentApplication"
             messageType="services:StudentServiceRequestMessage"/>
</variables>
<!--BPEL receive from a Web service call in-->
<receive name="receiveInput" partnerLink="client"
portType="tns:StudentFlowPlus"
     operation="initiate" variable="input" createInstance="yes"/>
<!-- retrieve Student ID using call out to session bean -->
<bpelx:exec name="RetrieveSSN" language="java" version="1.4">
 <![CDATA[
   try {
   // Retrieve element from scope
   Element element =
     (Element)getVariableData("input","payload","/StudentApplication");
   // Create an XMLFacade for the Student Application Document
   StudentApplication xmlStudentApp =
                   StudentApplicationFactory.createFacade(element);
   // Manipulate XML document through typed facade
   String email = xmlStudentApp.getEmail();
   // Use session bean to access external data source and lookup
   // the Student ID of the student based on his email.
   StudentHome cHome = (StudentHome) lookup("ejb/entities/Student");
   Student student = (Student) cHome.findByPrimaryKey( email );
   xmlStudentApp.setStudentId(Student.getStudentId());
   . . .  ]]> </bpelx:exec><!-Continue BPEL process -->
```

Figure 4: Appel bidirectionnel entre un composant Java et le processus BPEL

6. Conclusion

Nous avons proposé à travers cet article une des solutions techniques qui mettent en application différentes catégories d'agrégation de composants logiciels. Antérieurement, nous avons élaboré une description détaillée de ces agrégations et des métadonnées décrivant les composants à agréger (Masmoudi & al.,2005). Dans cet article, nous avons rappelé ces réalisations antérieures. Ensuite, nous avons décrit la structure, les outils et le mode d'exécution des processus BPEL par des exemples.

Comme son nom l'indique, BPEL est un langage d'exécution de processus métier. Ceci est un atout majeur dans le domaine des environnements d'apprentissage à distance. En effet, chaque module pédagogique d'un cours peut être décrit par un scénario pédagogique. Plusieurs outils éditent graphiquement les scénarios et les présentent sous un format XML standardisé tel IMS-LD. Cependant, rares sont ceux qui permettent l'exécution de ces scénarios comme le permet BPEL par les exemples décrits précédemment.

Techniquement, nos travaux avec ce langage nous prouvent qu'il est capable de concevoir des agrégations sur les quatre couches logicielles avec des composants de

différents degrés d'accessibilités. Cependant, nous pensons que le langage performe mieux dans l'agrégation par des boites noires, des boites grises et dans les coordinations des services web. Cette performance se justifie par le fait qu'à l'origine, ce langage visait à agréger des services web. Nous avons remarqué des limitations dans les couches de présentation (agrégation par collection) et dans les agrégations par fusion qui traitent des boites blanches. Nous planifions tester les spécifications WSRP[13] qui semblent garantir des agrégations d'interfaces usagers par les « portlets » invoqués à partir d'un processus BPEL.

Notre but ultime est de définir une méthodologie facilitant la tâche des ingénieurs responsables de construire des applications à partir de modèles de composants logiciels. Avec ce qu'on a présenté dans cet article, nous jugeons que les étapes de la méthodologie se clarifient de plus en plus et peuvent être matérialisées en ajoutant quelques règles de transition, des post-conditions et des pré-conditions. Voici une description sommaire des étapes anticipées. Nous commençons par la collecte des données sur les composants par la structure des métadonnées proposée. Ensuite, nous nous basons sur une combinaison des valeurs de trois de ces attributs pour définir la catégorie à laquelle appartiennent les composants à agréger. Une fois la catégorie identifiée, nous utilisons un ensemble d'activités de la structure du langage BPEL pour construire par modélisation le flux du processus de l'agrégat. Ce processus d'agrégation par collection, par coordination ou par fusion sera conçu, modélisé graphiquement, et exécuté par un serveur dédié. Il sera aussi suivi par une console appropriée pour contrôler graphiquement son état d'avancement et sa complétude.

Notre utilisation du langage BPEL ne s'arrête pas au niveau de l'agrégation effective des composants logiciels, mais nous pensons que même la méthodologie d'agrégation de composants dirigée par les modèles peut être conçue, modélisée et exécutée avec le langage BPEL et les outils le supportant. On parle à ce niveau d'un méta-processus (méthodologie) implémenté sous forme d'un processus BPEL qui assiste et dirige un ensemble de sous processus BPEL qui réalise concrètement les trois catégories d'agrégation. Cette méthodologie s'intègrera, une fois stabilisée, le système TELOS du projet LORNET pour servir les ingénieurs responsables de construire les systèmes d'apprentissage par des agrégations de composants.

Remerciements

Nous tenons à remercier le CRSNG[14] d'avoir financé le projet LORNET et l'équipe du laboratoire LICEF qui développe actuellement le système TELOS. Nous remercions les réviseurs d'avoir contribuer à l'amélioration de cet article.

[13] WSRP : Web Service Remote Portlets (www.oasis-open.org/committees/wsrp/)
[14] CRSNG : Conseil de recherches en Sciences Naturelles et en Génie du Canada

7. Bibliographies

BPEL Oracle Group. Oracle BPEL Process Manager, du site web à l'adresse http://www.oracle.com/technology/products/ias/bpel/index.html , 2005.

BPEL Oracle Group. BPEL Tutorials, du site web à l'adresse http://www.oracle.com/technology/products/ias/bpel/htdocs/dev_support.html#tutorials , 2004 – 2005

BPEL Oracle Quick Start, du site web à l'adresse http://www.oracle.com/technology/products/ias/bpel/pdf/orabpel-QuickStart.pdf, 2005

Eclipse BPEL, du site web à l'adresse http://www.eclipse.org/bpel/ , 2006

Ivers J., Clements P., Garlan D., Nord R., Schmerl B., Silva J. R. O., *Documenting Component and Connector Views with UML 2.0*, Software Engineering Institute, Avril 2004, Software Architecture Technology Initiative, Pittsburgh, disponible http://www.sei.cmu.edu/pub/documents/04.reports/pdf/04tr008.pdf (page consultée le 09 février 2005).

Lehmann, M., BPEL and J2EE: Bringing the Best of Both Together. Oracle Technology Network, 2005.

Masmoudi, A., Paquette, G., & Champagne, R. (2005). Agrégation de Composant logiciel Dirigée par les Métadonnées. Papier présenté à l'atelier OCM de la conférence INFORSID, Grenoble-France, mai 2005.

OASIS Group, Web Services Business Process Execution Language Version 2.0, from http://www.oasis-open.org/committees/download.php/16024/wsbpel-specification-draft-Dec-22-2005.htm, 2005.

Object Management Group, *Deployment and Configuration of Component-based Distributed Applications Specification*, Draft Adopted Specification ptc/03-07-02, juin 2003, disponible http://www.omg.org/docs/ptc/03-07-02.pdf (page consultée le 24 janvier 2005).

Paquette, G., Rosca. I, Mihaila S. and Masmoudi A. (2006 in press) TELOS, a service-oriented framework to support learning and knowledge Management in S. Pierre (Ed) E-Learning Networked Environments and Architectures: a Knowledge Processing Perspective, Springer-Verlag.

Szyperski C., Component Software: Beyond Object-Oriented Programming (2nd ed.). Addison-Wesley, 624 pages. 2002.

Selic, B., The pragmatics of model-driven development. Software, IEEE, 20(5), 19-25. 2003.

Villalobos J., Fédération de Composants : Une Architecture Logicielle pour la Composition par Coordination, Thèse de Doctorat, Université Joseph Fourier, Grenoble I, Juillet 2003.