Composition De Services

I. INTRODUCTION :

Parmi les huit caractéristiques déjà-vus concernant les services, on a dit qu'un service doit être composable, c à d qu'il peut être une pièce parmi un ensemble de pièces formant un nouveau service, ce dernier peut lui-même être regroupé avec d'autres, et ainsi de suite.

Un processus métier peut être vu comme étant une collaboration entre un ensemble de services, dans laquelle (collaboration) chaque service doit intervenir pour accomplir le travail des autres.

Deux paradigmes (orchestration et chorégraphie) sont utilisés pour réaliser une collaboration de services, ces deux paradigmes peuvent être cohabités dans le même PM.

1) Orchestration de services (Service Orchestration) [1][2] :

L'orchestration décrit l'interaction entre services, en incluant la logique affaire et l'ordre d'exécution des interactions. Ces interactions peuvent traverser des applications ou/et des organisations, et résulte dans un modèle de processus à multi étapes transactionnel et de longue durée.

Elle consiste à avoir un service «chef d'orchestre» qui joue le rôle d'intermédiaire entre les services en les appelant suivant un enchaînement prédéfini (script d'orchestration). Ce service est souvent appelé "moteur d'orchestration".

L'orchestration peut être vue comme une composition ascendante dans laquelle les services web appelés existent au préalable et n'ont pas besoin de savoir qu'ils font partie d'un processus plus gros. C'est uniquement le moteur d'orchestration connaît la logique de composition.

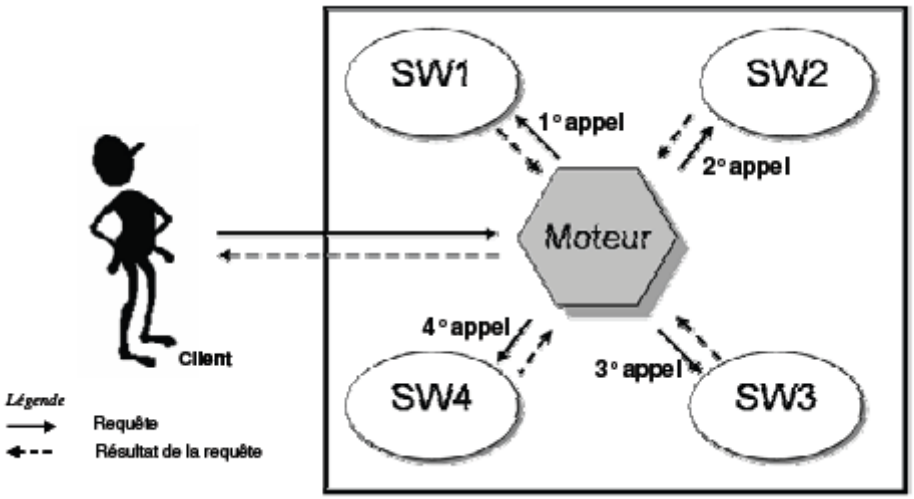


Figure 1: Vue générale de l'orchestration

Pour concevoir une orchestration de services web, il faut décrire les interactions entre le moteur d'exécution et les services web. Ces interactions correspondent aux appels effectués par le moteur, et aux actions proposées par les services web composants.

2) Chorégraphie (Service Choreography) [1][2]:

La chorégraphie trace la séquence des messages qui peuvent impliquer de multiples parties et de multiples sources, en incluant les clients, fournisseurs et partenaires. Elle est typiquement associée à l'échange de messages publics qui se produisent entre de multiples services web plutôt qu'à un processus métier spécifique qui est exécuté par une seule partie.

La collaboration de services dans une chorégraphie se passe d'une manière décentralisée sans utiliser un procédé principal, c'est-à-dire qu'elle dispense du rôle de chef d'orchestre, mais plutôt plusieurs procédés distribués (services participants à la collaboration) dont chacun d'eux doit être au courant de la logique du (ou des) processus auquel il appartient. Ainsi la logique de contrôle est supervisée par chacun des services intervenant dans la composition et l'exécution du processus est alors distribuée.

Exemple: un service doit par exemple savoir que « quand il reçoit un message d'un tel service, il doit attendre 50 secondes puis envoyer un message à un tel autre ».

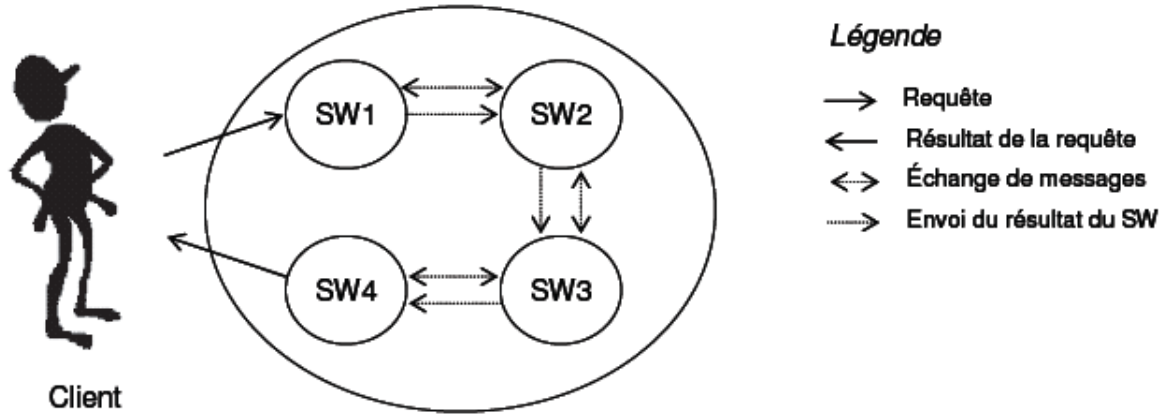


Figure 2: Vue générale d'une orchestration

Pour concevoir une chorégraphie, les interactions entre les différents services doivent être décrites. Ainsi la description de chaque service intervenant dans la chorégraphie inclut la description de sa participation dans le processus.

La chorégraphie est aussi appelée composition dynamique, du fait que l'exécution n'est pas régie de manière statique comme dans une l'orchestration. Dans une chorégraphie, à chaque pas de l'exécution (i.e. à chaque étape de la composition), un service web choisit le service web qui lui succède et implémente ainsi une partie de la chorégraphie. La composition de type chorégraphie n'est pas connue, ni décrite à l'avance.

La figure ci-dessous, décrit les deux paradigmes: L'orchestration se réfère à un processus exécutable, par contre la chorégraphie trace les séquences de messages entre parties et ressources.

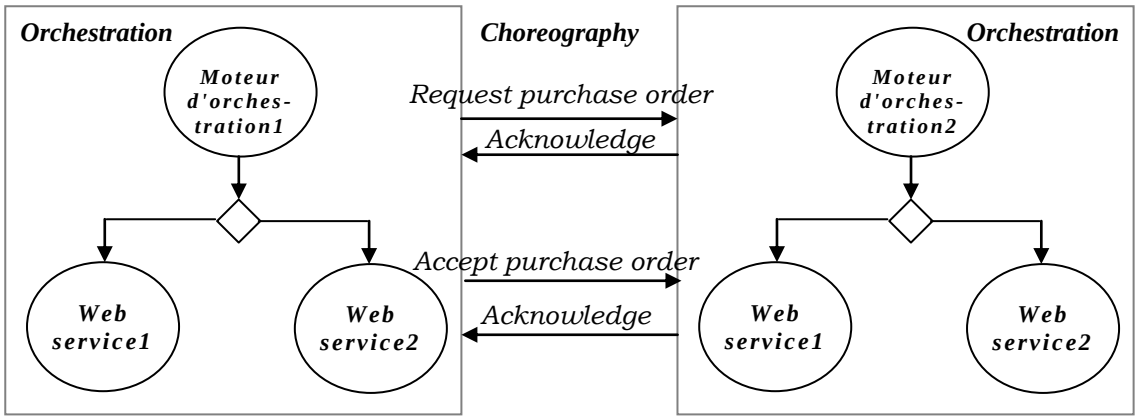


Figure 3: Orchestration vs Chorégraphie

II. STANDARDS DE DESCRIPTION DES PROCESSUS METIERS (COMPOSITION DE SERVICES):

Beaucoup de standards sont utilisés dans la description des processus métiers [2] [3] [4]:

1) XLANG :

C'est un format proposé par Microsoft, en 2001, pour représenter en XML l'orchestration d'activités qui constituent un processus métiers. XLANG est le format intermédiaire de stockage de l'environnement BizTalk server. Ces documents XLANG, ou encore appelé schedules : sont exécutés par le serveur BizTalk en phase de production. XLANG s'appuie sur WSDL en réutilisant un certain nombre de ses concepts.

Le document XLANG comprend :

a. ***Les actions XLANG*** : elles sont au nombre de quatre. Elles peuvent être élémentaires dans la représentation du *workflow* entre participants. Citons comme exemple : l'action *raise* pour signaler les défaillances, ou encore, *Operation* pour l'échange de message référençant le port d'un service donné.

b. ***Le contrôle XLANG*** : consiste en des commandes de contrôle de l'enchaînement de l'exécution des actions. Par exemple : la commande sequence permet l'exécution séquentielle d'une ou de plusieurs actions.

c. ***Les corrélations de messages*** : il est possible d'associer des champs d'information supplémentaires à un message d'une opération. Ces champs qu'ils soient simples ou complexes sont utilisés pour corréler des messages entre eux.

d. ***Le contexte*** : le contexte est un élément XLANG qui permet de définir des variables locales et des transactions liant les actions ou les combinaisons d'actions. Les variables locales sont constituées des définitions des corrélations et des références aux ports WSDL qui sont utilisées dans les actions.

e. ***Les transactions XLANG*** : une transaction préserve l'intégrité des données qu'elle manipule. Dans le cas de transactions dites longues, où les suspensions et les reprises d'exécution peuvent être beaucoup plus espacées dans le temps, les états même partiels doivent être sauvegardés pour éviter que leur verrouillage durable n'affecte d'autres transactions.

f. ***Les contrats*** : précisent comment lier deux ports différents des services XLANG. Pris ensemble avec les définitions des comportements XLANG, les contrats constituent la définition complète du workflow entre les différents services. La définition d'un contrat XLANG en XML consiste en général à importer la définition XLANG des deux services à relier et à spécifier comment les opérations de sortie de l'un aboutissent aux opérations d'entrée de l'autre.

2) WSFL (Web Service Flow Language):

Proposé par IBM, le WFSL est un langage de description de combinaison des services web de façon utile. Deux façons de combiner les services sont définies :

1) Des *modèles de circuit (Flow Model)*, où un circuit représente une séquence à exécuter pour qu'un processus métiers puisse être réalisé ;

2) Des *modèles globaux (Global Model)*, où la perspective est celle des régularités dans l'interaction entre les services web dans le contexte de collections de services typiques. Ces interactions sont réglées par des liens entre des services, avec un lien pour chaque

opération possible entre deux services dans une collection. Ce réglage peut être adapté soit à des interactions dans des hiérarchies, soit à des interactions collaboratives entre pairs.

3) BPML (Business Process Modeling Language)[1][4] :

BPML est un métalangage qui a été développé par l'organisation BPMI.org (Business Process Management Initiative), une organisation indépendante comprenant Intalio, Sterling Commerce, Sun, CSC, et d'autres. La spécification BPML fournit un modèle abstrait pour l'expression des processus métiers et des entités supportées.

BPML définit un modèle formel pour l'expression des processus métiers abstraits et exécutables qu'adresse tous les aspects des processus métiers, incluant les activités de complexités variées, les transactions de leur compensations, gestion des données, concurrent, gestion des exceptions et les sémantiques opérationnelles.

BPML peut aussi fournir une grammaire sous forme de Schéma XML pour permettre la persistance des échanges et des définitions à travers les systèmes hétérogènes et les outils de modélisations.

Ainsi un processus métier BPML est défini comme étant un enchaînement d'activités simples, d'activités complexes et de processus incluant une interaction entre participants dans le but de réaliser un objectif métier.

BPMI a identifié trois aspects fondamentaux des possibilités de BPML au moment de créer la spécification du langage :

- Le langage doit permettre de spécifier les transactions distribuées synchrones et asynchrones et les compensations, car le langage peut être utilisé par applications de mission critique.
- Le langage doit offrir des mécanismes fiables de sécurité, car il modélise des logiques métiers déployés sur Internet.
- Il doit contenir des méthodes de gestion de projet, car il est fait pour être employé dans les environnements intégrés de développement.

Les éléments de définition d'un procédé sont:

- Les messages (l'unité d'interaction du procédé).
- Les participants (des systèmes, des applications, des services web, des utilisateurs humains, des partenaires commerciaux, et d'autres processus), on distingue les participants statiques (où l'identité et le comportement sont connus à l'avance), et les participants dynamiques (ajoutés au moment de l'exécution).
- Les activités (l'unité d'exécution du procédé).
- Les transactions (lorsque les activités sont exécutées dans un contexte transactionnel) nous distinguons deux modèles de transactions coordonnées (pour les délais courts) et étendues (pour les délais longs).
- Les exceptions, pour les récupérations des erreurs.

4) WSCI (Web Service Choreography Interface)[4] :

WSCI, co-développé par BEA, Intalio, SAP et Sun est un langage de description basé sur XML dont l'objectif est de décrire les messages échangés entre les services web participants à des interactions de chorégraphie.

IL décrit l'interface dynamique du service web participant à un échange donné de message, en réutilisant les opérations définies dans l'interface statique. C'est-à-dire qu'il décrit comment les

opérations des services web (tel que ceux définies par WSDL) peuvent être chorégraphiés dans le contexte d'échange de messages dans lequel les services web y participent.

WSCI est la première étape permettant le mapping des services comme composants réalisant ces processus. WSCI décrit aussi comment la chorégraphie de ces opérations doit exposer les informations pertinentes, tel que la corrélation de message, la gestion des exceptions, la description des transactions et les capacités de la participation dynamique. Il ne suppose pas que les services web soient de différentes entreprises, comme dans le B2B; il peut être également utilisé pour décrire les interfaces des composants que représentent les unités organisationnelles internes ou d'autres applications dans une entreprise. WSCI n'adresse pas la définition de la conduite de l'échange de message ou la définition du comportement interne de chaque service web. WSCI décrit les interdépendances parmi les opérations des services web pour que n'importe quel client puisse:

1. Comprendre comment interagir avec un tel service dans le contexte du processus donné.
2. D'anticiper les comportements attendus d'un tel service à n'importe quel point dans le cycle de vie du processus.

Être capable de décrire l'interface dynamique d'un service dans le contexte d'un processus particulier qui permet de développer une forme abstraite de l'implémentation et de se focaliser sur le rôle joué des services web dans un tel processus.

Cependant, comme souligné par Dario Wiser, responsable du marketing produit chez Sun Microsystems « WSCI permet d'utiliser les services web au sein de processus complexes. Pourtant, WSCI n'est pas un concurrent des dialectes existants».

WSCI s'ajoute à WSDL et SOAP pour donner la description des messages, de leur ordre ainsi que de quelques autres attributs afin de permettre aux applications d'agir en collaboration avec succès.

Les interfaces de chorégraphie en WSCI sont caractérisées par:

- *Chorégraphie des messages* : une interface de WSCI décrit l'ordre dans lequel des messages peuvent être envoyés ou reçus dans un échange donné.
- *Frontières et compensation de transaction* : une interface de WSCI décrit les opérations déployées de manière transactionnelle. Cette information est importante pour les autres participants.
- *Gestion des threads* : l'interface WSCI décrit comment le service web contrôle les conversations multiples avec le même partenaire ou avec différents partenaires.
- *Connecteurs:* WSCI permet de tracer la consommation des opérations d'un service web pour produire des opérations à partir d'un autre service web afin d'établir clairement un modèle de l'échange global.
- *Participation dynamique* : l'interface de WSCI décrit comment l'identité du service cible (*target web service*) est choisie dynamiquement.

5) WS-CDL (Web Service Choreography Description Language)[2] [4] :

En 2004, la convergence de WSCI 24 et le WSCL 25 a donné lieu au WS-CDL. C'est un langage de description de la chorégraphie de services web qui se base sur le langage XML.

Il définit le comportement observable commun des entreprises participantes à la chorégraphie, dont l'échange de message est ordonné et synchronisé et résulte dans l'alignement de leur information partagée. Le contrat entre plusieurs participants est décrit par la chorégraphie. Cette dernière décrit la composition d'une perspective globale.

WS-CDL a pour objectif d'aboutir à la composition des collaborations interopérable entre n'importe quel type de participant, sans considérer la plateforme ou le modèle de développement utilisé.

6) WSCL (Web Services Conversation Language) [4] :

WSCL permet de décrire les conversations du niveau de la logique métier ou les processus publics basse sur la définition d'un service web. Les définitions de conversation en WSCL sont elles-mêmes des documents XML et peuvent donc être interprétées par des infrastructures de services web et des outils de développement.

La spécification WSCL est composée de quatre éléments principaux:

- Les *schémas* des documents XML, échangés au cours d'une conversation. Ces schémas ne font pas partie du document de spécification WSCL, ce sont des documents séparés référencés par une URL dans la spécification de la conversation ;
- Les *interactions*, modélisant les actions de la conversation comme des échanges de documents entre deux participants. Il existe cinq types d'interactions dans WSCL :
  - "Send" : émission d'un message,
  - "Receive" : réception d'un message,
  - "SendReceive" : émission puis réception d'un message,
  - "ReceiveSend" : réception puis émission d'un message,
  - "Empty" : ne contient pas de message à échanger, il est utilisé pour modéliser le début et la fin d'une conversation,
- Les transitions, spécifiant l'ordonnancement les relations entre les interactions. Chaque transition spécifie l'interaction source et l'interaction de destination, et éventuellement le type de message de l'interaction source ;
- La conversation, donne la liste de toutes les interactions et les transitions composant la conversation, ainsi que quelques informations additionnelles, comme le nom de la conversation, et l'interaction qui démarre la conversation, et celle qui la termine.

WSCL peut être employé en même temps que d'autres langages de description de service comme WSDL pour fournir des informations obligatoires de protocole pour les interfaces abstraites, ou pour indiquer les interfaces abstraites implémentées par un service concret.

III. BPEL4WS (Business Process Execution Language For Web Services):

En 2003, la spécification BPEL4WS1.1 a émergé après la convergence de deux spécifications à savoir le *WSFL* et le *XLANG*, suite aux efforts unis de différents organismes: *BEA, IBM, SAP, Siebel Systems* et *Microsoft*. Il combine les caractéristiques d'un langage de processus structuré par bloc (*XLANG*) avec ceux d'un langage de processus basé sur les processus métiers (*WSFL*).

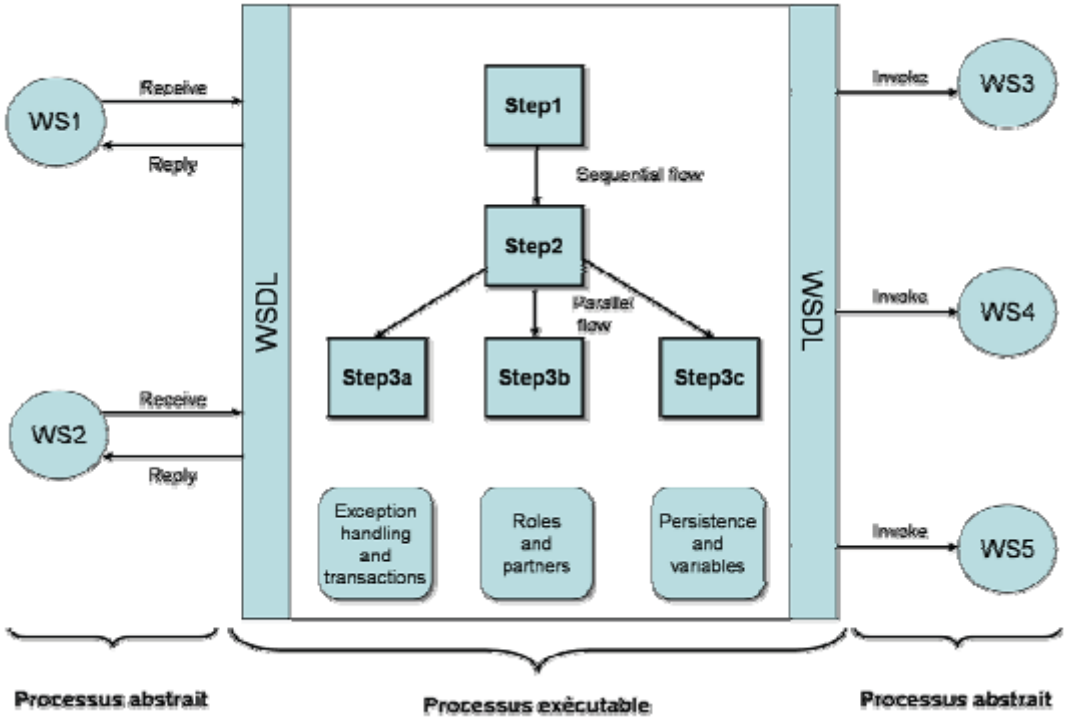
BPEL4WS forme une couche au-dessus de *WSDL* et fournit une notation *XML* et une sémantique pour la spécification du comportement des processus métiers basés sur les services web. Un processus BPEL4WS est défini en terme de ses interactions avec ces partenaires. Un partenaire peut fournir des services au processus, nécessiter des services des processus, ou encore, participer à l'interaction avec le processus [1][5].

Remarque : Une nouvelle version du BPEL, a été publiée le 31 Janvier 2007, appelé *WSBPEL*.

BPEL4WS permet de modéliser deux types de procédé:

1) Le processus abstrait (procédé abstrait) :

Ce processus spécifie les messages échangés entre les différentes parties (services web composant) sans indiquer le comportement de chacune d'elles. Il peut être relié à une composition de type chorégraphie.



Le flot de processus avec BPEL4WS

Le grand avantage de BPEL est la possibilité de décrire les interactions entre les logiques métiers des différentes entreprises.

2) Processus exécutable:

Ce processus décrit le comportement actuel d'un participant dans une interaction métier. Il permet de spécifier l'ordre d'exécution des activités, le partenaire concerné, les messages échangés entre ces partenaires, et les mécanismes de traitement des erreurs et des exceptions. En d'autres termes, il s'agit du moteur de l'orchestration donnant une représentation indépendante des interactions entre les partenaires [3].

1) Les éléments de base:

Trois éléments principaux permettent à BPEL4WS de gérer le flot de processus dans le processus exécutable:

a. **Les partenaires** (***Partners***)**:** Les partenaires sont différents services web invoqués dans le processus. Ils ont chacun un rôle spécifique dans un processus donné. Chaque partenaire est décrit par son nom, son rôle (en tant que service indépendant), et son rôle dans le processus (définitions de deux partenaires dans l'exemple: lignes 2 à 11). Le fait de décrire deux niveaux de rôle permet à chaque partenaire d'avoir une vie indépendante des compositions dans lesquelles il intervient.

b. **Les variables (espaces de stockage) (Containers):** Les espaces de stockage permettent la transmission des données. Le flot de processus BPEL4WS permet que ces données soient cohérentes à travers les messages échangés entre les services web. Un message peut être un message d'appel (*invoke*), de réponse (*reply*) ou d'attente (*receive*).

c. **Les activités (transactions):** Les activités sont utilisées pour invoquer les autres services. La structure des différentes activités peut être, soit séquentielle soit parallèle. Si l'activité est parallèle, alors plusieurs services peuvent être invoqués en même temps.

2) Exemple de fichier BPEL4WS "Processus MyMeteo":

Examinons l'exemple suivant qui illustre un extrait du processus "myMeteo".

```
1  <process name="myMeteo">
2      <partners>
3          <partner name="geoIPService"
4                   serviceLinkType="geoIPLink"
5                   myRole="geoIP"
6                   partneRole="geoIPContext"/>
7          <partner name="globalWeather"
8                   serviceLinkType="globalWeatherLink"
9                   myRole="weather"
10                  partnerRole="globalWeatherService"/>
11     </partners>
12     <containers>
13         <container name="getLocalisation" messageType="getGeoIPContextSoapIn"/>
14         <container name="getWeatherContainer" messageType="getWeatherSoapIn"/>
15     </containers>
16     <flow>
17         <invoke partner="geoIPService"
18                 portType="GeoIPServiceSoap"
19                 operation="GetGeoIPContext"
20                 inputContainer="getLocalisation" />
21         <invoke partner="globalWeather"
22                 portType="GlobalWeatherSoap"
23                 operation="GetWeather"
24                 inputContainer="getWeatherContainer" />
25     </flow>
26 </process>
```

Définition du processus exécutable de la demande du service de météorologie *myMeteo* à l'aide de BPEL4WS.

A partir de la Figure, nous pouvons mettre en relief la syntaxe de BPEL4WS en décrivant le processus myMeteo (élément process, élément racine du document BPEL4WS, ligne 1). Les différents partenaires geoIPService et globalWeather) sont en premier lieu définis (élément partners, lignes 2 à 11). Ces deux acteurs interagissent avec le service qui orchestre la composition par l'intermédiaire de deux messages nommés getLocalisation et getWeather (élément containers, lignes 12 à 15). L'élément containers référence le type de message (attribut messageType, lignes 13 et 14) contenus dans les descriptions WSDL des services web composants.

Ainsi, le service web qui orchestre connaît les types de données manipulées. Le flot de processus (élément flow, lignes 16 à 25) se compose tout d'abord de l'appel du partenaire geoIPService (élément invoke, lignes 17 à 20) puis de l'appel du partenaire globalWeather (élément invoke, lignes 21 à 24). L'élément invoke comprend les accès aux ports (attribut portType, lignes 18 et 22) et les opérations (attribut operation, lignes 19 et 23) référençant la description WSDL des servicesWeb composants. La Figure suivante n'illustre qu'un extrait de la description du processus exécutable myMeteo. BPEL4WS intègre aussi les flots de données dans la description du processus en reliant les sorties d'un service avec les entrées du service lui succédant.

BPEL4WS est le premier langage de composition de services Web adopté par la communauté des services Web. Ceci est principalement dû au fait que ce langage possède une grande expressivité dans la définition du processus exécutable. L'inconvénient principal de BPEL4WS est que la définition du processus est rigide. Si une activité du processus échoue, le processus dans son intégralité échoue. Aucun retour en arrière et aucune alternative au processus ne sont possibles. De même, lors de la description du processus exécutable, la définition des flots de données ne permet pas de connecter des services dont les entrées/sorties ne correspondent pas exactement. BPEL4WS ne prévoit pas de mécanisme de transformation de données.

3) Structure d'un fichier BPEL:

Un fichier BPEL est globalement structuré comme suit:

```
<process>
        (superior attributes)
        <partners> …  </partners>
        <variables> …  <variables>
        <correlationSets>  … </correlationSets>
        <faultHandlers>  …  </faultHandlers>
        <compensationHandlers> … </compensationHandlers>
        <eventHandlers> … </eventHandlers>
        (activities)*
</process>
```

a. Attributs supérieurs :

Cette section correspond à la définition des attributs de description du fichier comme le nom et les espaces de noms à utiliser pour le procédé.

b. Partenaires (partners, partnerLink):

Un lien de partenaire (partnerLink) correspond au service avec lequel le procédé échange des informations. Le lien de partenaire représente la relation de conversation entre deux procédés partenaires. Chaque lien de partenaire est typé par un partnerLinkType, il est chargé de définir le rôle que joue chacun des deux partenaires dans une conversation.

c. Variables (variables, containers):

Le procédé dans BPEL a un état, cet état est maintenu par des variables contenant des données. Ces données sont combinées afin de contrôler le comportement du procédé. Elles sont utilisées dans les expressions et les opérations d'affectation. Les expressions permettent d'ajouter des conditions de transition ou de jointure au flot de contrôle. L'affectation (assignment) permet de mettre à jour l'état du procédé, en copiant les données d'une variable à une autre ou en introduisant de nouvelles données en utilisant les expressions.

Dans BPEL il n'y a pas de flot de données, BPEL se sert des variables pour passer une donnée d'une activité à une autre, à l'aide de l'affectation.

d. Corrélation (correlationSets):

Cette partie permet d'utiliser les paramètres de corrélation de message pour maintenir les différentes conversations avec les partenaires.

e. Gestion des erreurs (faultHandlers):

Sert à déterminer les activités à exécuter en réponse à une erreur survenue lors d'une invocation d'un service.

f. Recouvrement (compensationHandler):

Cette partie sert à reverser le travail effectué par une activité annulée ou aboutie à une erreur. BPEL fournit un mécanisme qui garde traces (historique) des opérations effectuées par une transaction.

g. Activités (sequence, flow):

Le procédé dans BPEL est constitué d'activités liées par un flot de contrôle. Ces activités peuvent être basiques ou structurées.

Les activités basiques sont :

- <invoke>: pour invoquer une opération dans un service web.
- <receive>: pour attendre un message d'une source externe.
- <reply>: pour répondre à une source externe.
- <wait>: pour attendre un certain temps.
- <assign>: pour copier les données d'une place à l'autre.
- <throw>: pour lancer une erreur d'exécution.
- <terminate>, <exit>: pour terminer l'instance de service en entier.
- <empty>: ne rien faire (utile pour la synchronisation des activités parallèles).

Les activités structurées sont composées d'autres activités basiques et structurées. Les types d'activités structurées sont :

- <sequence>: pour définir un ordre d'exécution.
- <switch>, <if>: pour l'acheminement conditionnel.
- <while>, <repeatuntil>: pour les boucles.
- <foreach>: branchement multiple.
- <pick>: pour attendre l'arrivée d'un événement
- <flow>: pour l'acheminement parallèle.
- <scope>: pour regrouper les activités afin qu'elles soient traitées par le même gestionnaire d'erreur.
- <compensate>: pour invoquer les activités de compensation par le gestionnaire d'erreur, pour défaire l'exécution déjà complétée d'un regroupement d'activité.

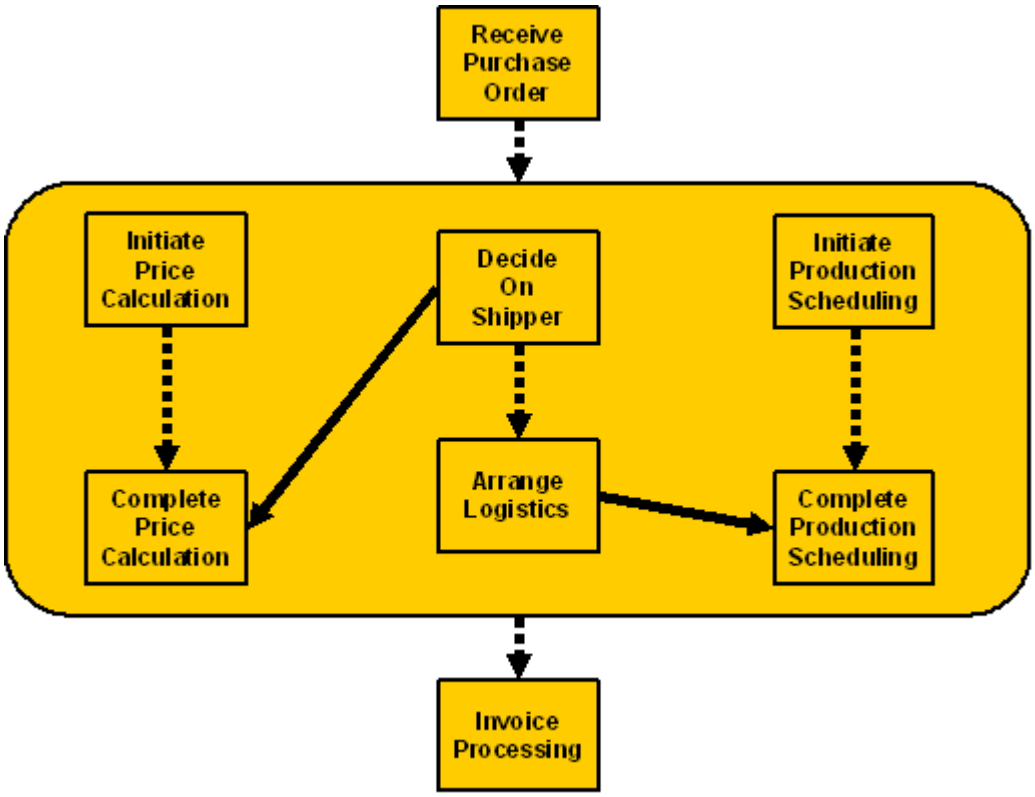
Annexe :

Example 2 (english) [5] :

Before describing the structure of business processes in detail, this section presents a simple example of a WS-BPEL process for handling a purchase order. The aim is to introduce the most basic structures and some of the fundamental concepts of the language.

The operation of the process is very simple, and is represented in Figure 1: Purchase Order Process Outline. Dotted lines represent sequencing. Free grouping of sequences represents concurrent sequences. Solid arrows represent control links used for synchronization across concurrent activities. Note that this is not meant to be a definitive graphical notation for WS-BPEL processes. It is used here informally as an aid to understanding.

On receiving the purchase order from a customer, the process initiates three paths concurrently: calculating the final price for the order, selecting a shipper, and scheduling the production and shipment for the order. While some of the processing can proceed concurrently, there are control and data dependencies between the three paths. In particular, the shipping price is required to finalize the price calculation, and the shipping date is required for the complete fulfillment schedule. When the three concurrent paths are completed, invoice processing can proceed and the invoice is sent to the customer.



Références :

[1] Parameswaran Seshan « Process-Centric Architecture for Enterprise Software Systems » Auerbach Publications 2010

[2] Céline Lopez-Velasco: « Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation». Thèse de doctorat soutenue en 2008 Université de Joseph Fourier.

[3] Wikipedia : «https://en.wikipedia.org/wiki/Service_choreography».

[4] A Hands-on Introduction to BPEL «https://www.oracle.com/technical-resources/articles/matjaz-bpel.html».

[5] OASIS Standard : « Web Services Business Process Execution Language Version 2.0» accessible sur «http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html » .