REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTRE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE LARBI BEN MHIDI D'OUM EL BOUAGHI

FACULTE DES SCIENCES EXACTES ET SNV

DEPARTEMENT DE MATHEMATIQUES ET INFORMATIQUE

Mémoire :

Présenté pour l'obtention du diplôme de Master en Informatique

Spécialité :

Architechture Distribuée

Proposition d'une approche de Monitoring des Qualités de services (QoS) des services web composés

Présenté par :

Titi Ouahiba

Devant le jury :

Président :	Dr. HAMRI SALAH
Encadreur :	Dr. BENABOUD Rohallah
Examinateur :	Mr. MEROUANI HAMZA

Promotion juin 2017

Résumé :

Les services web sont des fonctionnalités logicielles publiées et accessible par internet, ils répondent généralement à des besoins fonctionnels et non fonctionnels, les besoins non fonctionnels des services web s'expriment à travers la qualité de service (QdS). La QdS est décrite par des paramètres tels que la performance, la disponibilité et la fiabilité. Ces paramètres doivent être définis afin de pouvoir distinguer les services web similaires. Le monitoring des QdS consiste à surveiller et mesurer les paramètres des QdS d'un service web afin de vérifier la conformité des valeurs de QdS publiées par le fournisseur de service.

Dans ce mémoire, nous proposons une approche pour le monitoring des QdS des services web, nous considérons le problème de mesure des QdS d'un ensemble des services web composés au sein d'un processus métier, nous illustrons notre approche sur deux études de cas dont la spécification est donné par le langage de composition des services BPEL.

Mots clés : service web, qualité de service, monitoring, composition des services, BPEL.

Abstract :

Web services are published software features that are accessible via the Internet, they generally meet functional and non-functional requirements, the non-functional requirements of web services are expressed through quality of service (QoS). The QoS is described by parameters such as performance, availability and reliability. These parameters must be set in order to distinguish similar web services. QoS monitoring consists of surveillance and measuring the QoS parameters of a web service to verify the compliance of the QoS values published by the service providers.

In this work, we propose an approach for the web services QoS monitoring, we consider the problem of QoS measurement of a set of web services composed in a business process, we illustrate our approach on two study cases developed under the web services composition language BPEL.

Keywords : web service, quality of service, monitoring, web services composition, BPEL.

ملخص :

تعتبر خدمات الويب من الوظائف البرمجية المتوفرة عبر شبكة الإنترنت، والتي عادة ما تستجيب لمتطلبات وظيفية وغير وظيفية ، ويتم التعبير عن الاحتياجات غير الوظيفية لخدمات الويب من خلال جودة الخدمة، توصف جودة الخدمة بعدة مقاييس مثل الفعالية والتوافر والموثوقية.هذه المقاييس يجب أن تكون معرفة من أجل تمييز خدمات الويب المتماثلة. رصد نوعية الخدمة هي عملية مراقبة وقياس معايير جودة الخدمة لخدمات الويب للتحقق من دقة قيم نوعية الخدمة التي نشرت من قبل مقدمي الخدمات.

في هذه المذكرة، نقترح نهجا لرصد جودة الخدمة لخدمات الويب، ونأخذ بعين الاعتبار مشكلة قياس جودة الخدمة لمجموعة من خدمات الويب المركبة، كما نقوم بتطبيق النهج المقترح على مثالين مقدمين على لغة البرمجة لخدمات الويب المركبةBPEL .

الكلمات المفتاحية : خدمات الويب، جودة الخدمة، الرصد، خدمات الويب المركبة، BPEL

Remerciements :

Avant tous, louange à ALLAH de m'avoir donné le courage, la force, la volonté, et la patience pour terminer ce modeste travail.

Je tiens à exprimer d'abord toute ma gratitude à mon encadreur Dr BENABOUD Rohallah de l'université Larbi Ben Mhidi pour m'avoir proposé ce travail, pour son encadrement, son écoute, ses élucidations, ses conseils, ses directives et encouragements qu'il m'a afflués. Ainsi que tous les enseignants qui m'ont suivi durant mes cinq années d'étude.

Je remercie Dr HAMRI Salah de m'avoir fait le bonheur de présider le jury de ma soutenance.

Je remercie aussi Mr MEROUANI Hamza pour l'intérêt qu'il a porté à ce travail en acceptant d'être examinateur.

Enfin, j'adresse mes remerciements à toute personne ayant contribué de prés ou de loin à la concrétisation de ce travail.

Dédicace :

Je dédie ce modeste travail synonyme et concrétisation de tous mes efforts fournit dans ces dernières années :

A mon symbole de sacrifice et d'affection, à celui que j'estime le plus : à ma chérie mère et mon cher père.

A ma sœur AMINA et ses petits SKANDER et MAHER, a mes sœurs HANANE, KHADIDJA, et MANEL et mon petit frère OUASSIM.

Spécialement à mon fiancé AMMAR pour son encouragement, sa patience et son aide.

A Mon âme sœur SAFA, et mes adorables amies CHAIMA, RAWNA, AMIRA et IMENE, et tous les amis qui je n'ai pas cités, en particulier les étudiants de ma promotion Master II Architecture Distribuée à l'université d'Oum el bouaghi.

A la fin, je veux dédie ce modeste travail à tous les enseignants et les enseignantes qu'ils m'aider pour arriver ici depuis le début de mes études.

Table des matières

Liste des figures :

Liste des tableaux :

Introduction générale

Introduction générale :

Les systèmes informatiques, notamment le Word Wide Web, deviennent de plus en plus complexes et hétérogènes en raison de la diversité des fournisseurs, des clients et de leurs besoins, avec un marché plus exigeant et évolué. Dans ce contexte, l'architecture orientée services (SOA) peut être considérée comme étant une solution très efficace pour l'évolutivité et l'adaptabilité de ces systèmes, en améliorant leur qualité et en simplifiant leur intégration dans l'infrastructure de l'entreprise [30].

L'architecture orientée service est basée sur l'utilisation de composants réutilisables et distribués appelés services web, qui sont des fonctionnalités logicielles publiées et accessible via les protocoles d'internet; leur architecture par composants permet à une application de faire l'usage d'une fonctionnalité située dans une autre application [19].

Bien que les services web constituent une révolution dans le Word Wide Web, leur utilisation en tant qu'entités individuelles limitent leurs capacités, l'entreprise est amenée à composer un ensemble de services afin d'accomplir des buts complexes non réalisables par un seul service web. On parle ainsi de la composition des services. Cette dernière consiste à combiner les fonctionnalités de plusieurs services web au sein d'un même processus métier (*en anglais, business process*) ou service web composé dans le but de répondre aux demandes complexes du client [1].

Puisqu'ils sont destinés à être découverts et utilisés par d'autres applications web, les services web doivent être décrits et entendus en termes de propriétés fonctionnelles et non fonctionnelles, on parle ici de la qualité de service (QdS). La QdS est décrite par les paramètres standards [2] décrivant la performance et d'autres paramètres plus spécifiques tels que la disponibilité et la Scalabilité. Ces paramètres doivent être bien définis afin de pouvoir distinguer les services web qui sont fonctionnellement similaires.

Afin d'avoir une vision claire de la performance des services web et de leurs paramètres de QdS, l'exécution des services web doit être surveillée une fois que ces derniers deviennent opérationnels ; cette activité est appelée le Monitoring de QdS. Le Monitoring est l'une des tâches les plus importantes dans le cycle de vie du service Web, car il permet de vérifier la conformité des valeurs de QdS publiées par le fournisseur, et d'obtenir une vision précise de la façon dont les services se comportent dans leurs environnements opérationnels.

La composition des services se présente comme un paradigme fondamental dans la technologie des services Web. De ce fait, le besoin de Monitoring d'un service composé et de sa qualité s'impose dans le monde du Web. Le mécanisme de Monitoring doit prendre en compte la mesure des valeurs des QdS des différents services web impliqués dans la composition pour ensuite calculer les valeurs de QdS du service composé (processus métier) en utilisant différentes métriques.

Dans notre travail, nous proposons une approche pour le Monitoring des QdS des services web composés, cette approche consiste à utiliser le paradigme orienté aspect pour mesurer la qualité de chaque service participé dans le processus métier pour enfin avoir un monitoring global des services composés. Afin de montrer la pertinence de notre approche, nous l'appliquons sur deux études de cas.

Notre mémoire est organisé, en quatre chapitres, comme suit :

Chapitre 1 : nous commençons par introduire d'une façon générale l'architecture orientée service, la technologie des services, leurs fonctionnement ainsi que leurs avantages. Puis nous présenterons la notion de QdS et son intérêt pour les services Web.

Chapitre 2 : nous présentons dans ce chapitre un état de l'art sur les différentes techniques de Monitoring des QdS des services Web afin de bien situer notre mécanisme proposé.

Chapitre 3 : nous présentons notre approche : l'architecture globale, les différents modèles de composition des services web, ainsi que les métriques proposées pour le Monitoring de la QdS.

Chapitre 4 : nous nous intéressons à l'application développée ; nous présenterons d'abord les différentes technologies utilisées, ensuite l'application de notre approche de Monitoring sur études de cas différentes.

Pour conclure notre mémoire, nous terminerons par une conclusion générale et quelques perspectives de recherche de ce travail.

Chapitre 1 : les Services Web

1. Introduction :

De nombreuses organisations tournent actuellement vers des architectures à base des services web pour le développement et l'intégration des applications ou des systèmes d'information [9]. Nous présentons dans ce chapitre la notion de l'Architecture orientée service et service web tout en illustrant ses technologies, nous montrons aussi la notion de QdS et ses exigences dans les web services.

2. Architecture Orientée Service (SOA) :

2.1. Définition :

On peut dire que l'architecture orientée service (en anglais : Service Oriented Architecture SOA) est une architecture logicielle s'appuyant sur des services simples en interaction :

« *L'AOS est une approche architecturale permettant la création des systèmes basés sur une collection de services développés dans différents langages de programmation, hébergés sur différentes plates-formes avec divers modèles de sécurité et processus métier [2]* » :

- Chaque service représente une unité autonome de traitement et de gestion de données, communiquant avec son environnement à l'aide de messages. Les échanges de messages sont organisés sous forme de contrats d'échange.
- L'idée maîtresse de l'architecture orientée service est que tout élément du système d'information doit devenir un service identifiable, documenté, fiable, indépendant des autres services, accessible, et réalisant un ensemble de tâches parfaitement définies.
- Un des avantages des SOA réside dans le fait que les applications réparties n'ont plus besoin d'un système de middleware réparti commun pour communiquer, mais seulement des protocoles et des technologies de communication interopérables sur Internet [9].

2.2. Caractéristiques de SOA :

L'architecture orientée service est caractérisée par [12] :

- Un couplage faible entre les services : implique qu'un service n'appelle pas directement un autre service.
- La réutilisation de service.
- L'indépendance par rapport aux aspects technologiques : c.-à-d. Les services avec cette architecture sont indépendants de ses plates-formes.
- La découverte des services disponibles.
- La mise à l'échelle est rendue possible grâce à la découverte et à l'invocation des nouveaux services lors de l'exécution.

2.3. Les acteurs de SOA :

L'architecture SOA est basée sur trois acteurs, à savoir, le fournisseur de services, le client de services, et l'annuaire de publication [12] :

- ✓ Le fournisseur de services (Service Provider) : il met le service web en application et il le rend disponible pour tout le monde sur internet.
- ✓ Le client de services (Service Requester) : un consommateur qui fait la demande d'un service web bien précis pour répondre à ses besoins.
- ✓ L'annuaire de publication (Service Registry) : fournit un endroit où le consommateur peut trouver des nouveaux services web et le fournisseur dispose une description pour des nouveaux services web.

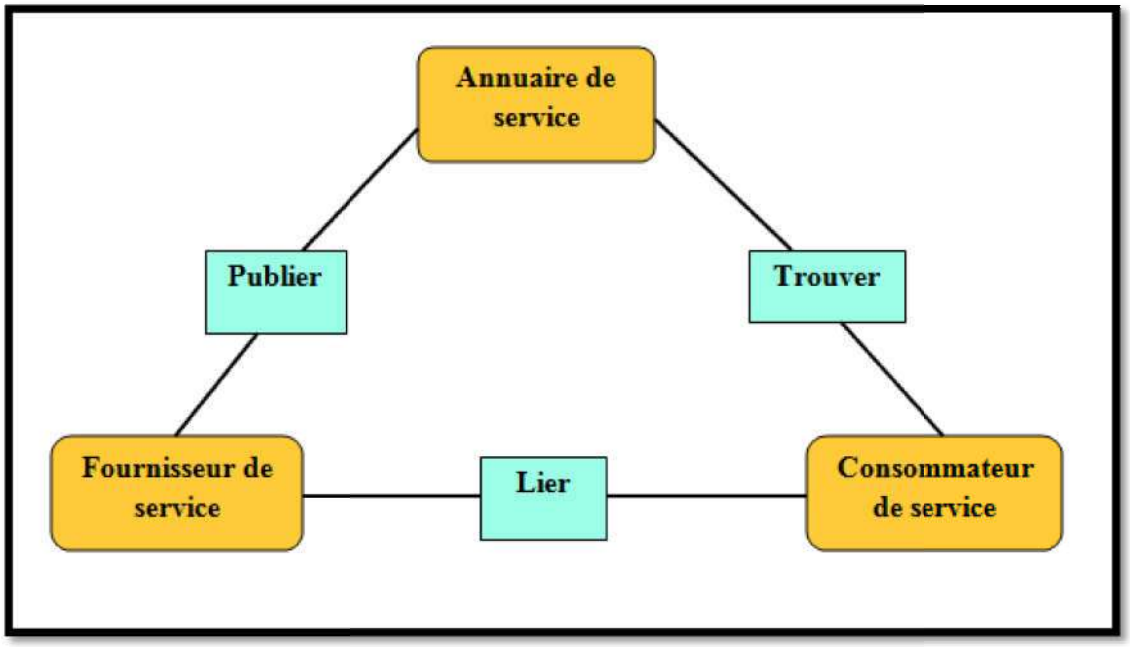

Figure 1 : Les acteurs principaux de SOA [12].

3. Les services Web :

3.1. Définition :

Plusieurs définitions des services Web ont été mises en avant par différents auteurs, chaque auteur définit les services web par des caractéristiques technologiques distinctives. Nous avons choisi la définition la plus générique et lisible : la définition de W3C[1] :

« *Un service web est un système logiciel conçu pour supporter les interactions entre application à travers le réseau. Les services web offrent un moyen standard d'interopérabilité entre différentes applications qui s'exécutent sur une variété de machine/plateforme. Ils sont caractérisés par leur grande interopérabilité et extensibilité, ainsi qu'une description interprétable/compréhensible automatiquement par la machine grâce au standard XML. Ils peuvent être combinés d'une façon faiblement couplée afin de réaliser des opérations complexes. Les programmes offrant des services simples, peuvent interagir ensemble afin de mettre en place des services sophistiqués avec des valeurs ajoutées* [2]».

D'une façon plus claire, on peut dire qu'un service web est un programme informatique permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués [3]. Il s'agit donc d'un ensemble de fonctionnalités exposées sur internet ou sur un intranet, par et pour des applications ou machines, sans intervention humaine, et en temps réel.

[1]World Wide Web Constrium, http://www.w3c.org/

3.2. Caractéristiques des Web services :

A partir de la définition donnée précédemment, nous pouvons distinguer différentes caractéristiques des services web tel que l'interopérabilité, l'extensibilité, l'accessibilité via le réseau, …etc. Les principaux caractéristiques des Web services sont [36] :

- Composantes réutilisables.
- Interface asynchrone utilisant des technologies indépendantes des plates-formes.
- Composante logicielle légèrement couplée à interaction dynamique.
- Les bases permettant de construire des systèmes distribués et ouverts sur internet.
- Capacités d'interagir avec d'autre composantes logicielles via des éléments XML et utilisant des protocoles internet.
- Capacités d'interfaces et associations d'être publiées, localisées et invoquées via XML.

3.3 Cycle de vie et fonctionnement des services web:

L'architecture d'un service web se repose sur les trois acteurs indiqués précédemment : le fournisseur de service, le client de service et l'annuaire de publication. La figure 2 montre d'une façon plus détaillée le cycle de vie de l'utilisation de service web entre ces trois acteurs :

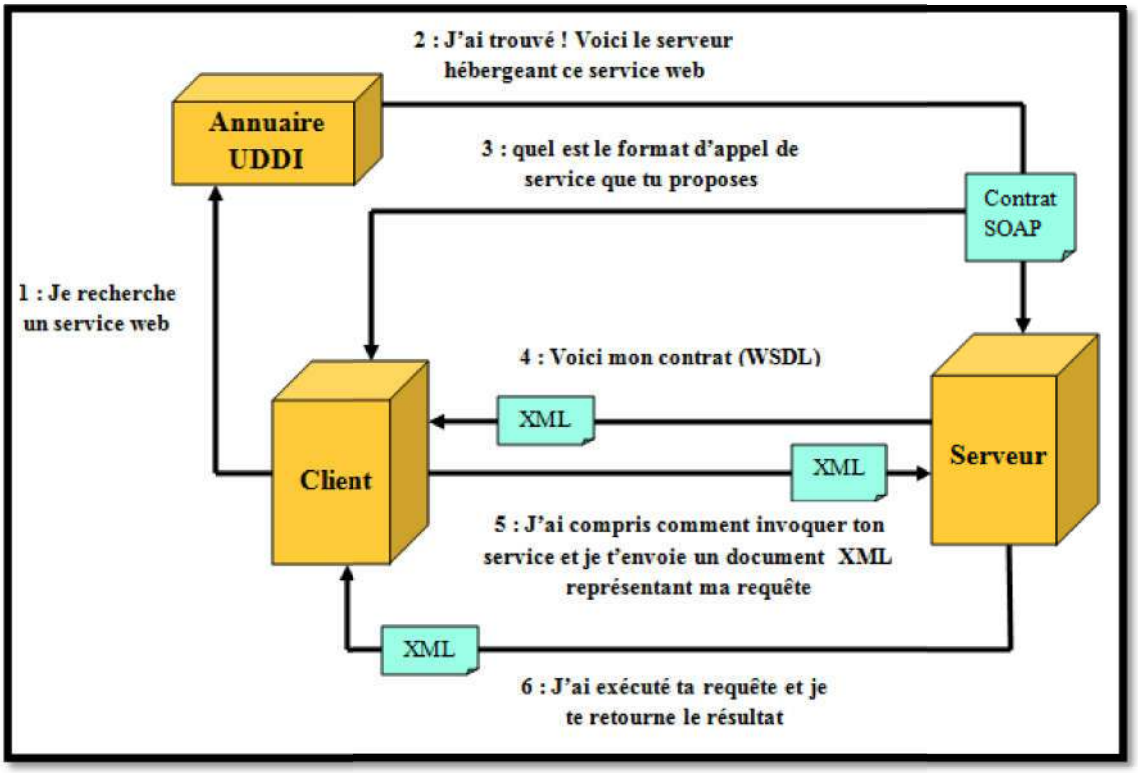

Figure 2 : Cycle de vie de l'utilisation de service web [9]

Pour expliquer le fonctionnement des services web, il convient de distinguer plusieurs étapes [4] :

Recherche dans un annuaire UDDI : le client cherche un service particulier, il s'adresse à un annuaire qui va lui fournir la liste des prestataires habilités à satisfaire sa demande. L'annuaire UDDI peut être comparé à un moteur de recherche sauf que les documents sont remplacés par des services.

Recherche de l'interface du composant à contacter : une fois la réponse reçue (en XML) de l'annuaire, le client va chercher à communiquer via une interface. Cette interface décrite en langage WSDL fournit l'ensemble des services disponibles. Elle offre la possibilité de vérifier que le contrat correspond bien aux besoins demandés.

Invocation du service : le client doit maintenant passer les paramètres attendus par le service et assurer la communication avec le serveur. L'outil utilisé pour cela est le Proxy, c'est l'objet qui permet la communication entre le client et le serveur. Il est généré par le client en utilisant l'interface WSDL.

3.4. Les Technologies des services Web :

L'infrastructure des services web s'est concrétisée autour de trois spécifications considérées comme des standards, à savoir WSDL, SOAP et UDDI :

3.4.1. WSDL (Web Service Description Language) :

Permet de donner une description au format XML des Web Services en précisant les méthodes pouvant être invoquées, leur signature et le point d'accès (URL[2], port, etc.)[3].

Structure d'un document WSDL :

La structure d'un document WSDL se compose de deux parties importantes [9]:

La partie abstraite : décrit les messages envoyés et les messages reçus, et les associés d'exploitation d'un modèle d'échange de messages avec un ou plusieurs messages.

La partie concrète : précise les détails du format de transports et de fil pour une ou plusieurs interfaces, un port associe une adresse réseau avec une liaison et un service qui regroupe les points de terminaison qui implémente une interface commune.

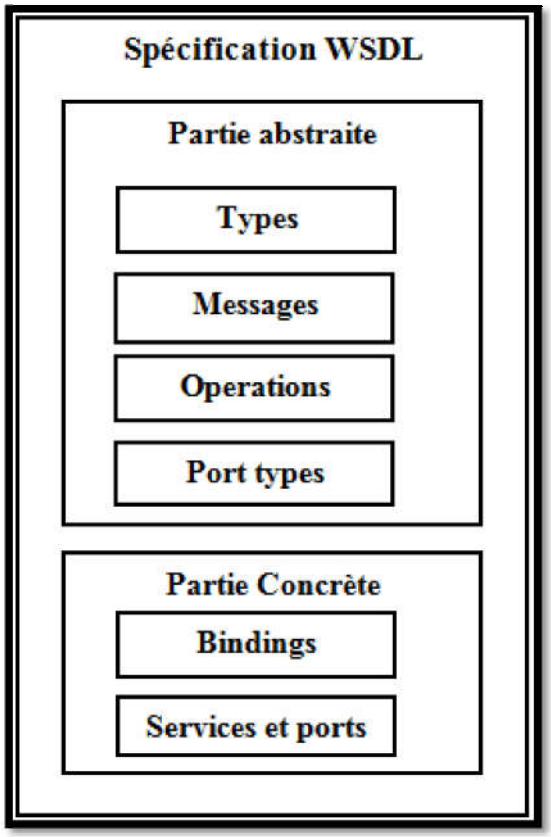

Figure 3 : Structure d'un document WSDL [9]

Le document WSL est formé des six éléments suivants :

- ✓ L'élément « type » : définition de type de données utilisées lors de l'échange.
- ✓ L'élément « message » : représentation abstraite de contenu d'un message (données à transmettre, valeurs de retour).

[2]Uniform Resource Locator

- ✓ L'élément « portType » : définition d'un ensemble d'opérations offertes par un service web.
- ✓ L'élément « binding » : définition d'un protocole de transport et le format des messages.
- ✓ L'élément « service » : ensemble de ports (lien URL) associés à une opération donnée.
- ✓ L'élément « port » : adresse d'une liaison définissant le point d'accès associé à un service.

3.4.2. SOAP (Simple Object Access Protocol) :

SOAP est un protocole d'invocation de méthodes sur des services distants. Basé sur XML, c'est un format de communication pour assurer la communication de machine à machine. Le protocole permet d'appeler une méthode RPC[3] et d'envoyer des messages aux machines distantes via HTTP [4].

Ce protocole est très bien adapté à l'utilisation des services web car il permet de fournir au client une grande quantité d'informations récupérées sur un réseau de serveurs tiers [4].

SOAP permet donc l'échange d'information dans un environnement décentralisé et distribué, comme Internet par exemple. Il permet l'invocation de méthodes, de services, de composants et d'objets sur des serveurs distants et peut fonctionner sur de nombreux protocoles (des systèmes de messagerie à l'utilisation de RPC).

SOAP utilise principalement les deux standards HTTP et XML [2]:

- ✓ HTTP comme protocole de transport des messages SOAP. Il constitue un bon moyen de transport en raison de sa popularité sur le web.
- ✓ XML pour structurer les requêtes et les réponses, indiquer les paramètres des méthodes, les valeurs de retours, et les éventuelles erreurs de traitements.

- ***Structure d'un message SOAP :*** un message SOAP contient les quatre parties suivantes:

Une enveloppe : définit le cadre pour décrire ce qui est dans le message et comment le traiter [36].

Les règles d'encodage (header) : offrent des mécanismes flexibles pour étendre un message SOAP sans aucune préalable connaissance des parties communicantes. Les

[3]Remote Procedure Call : protocole réseau permettant de faire des appels de procédures à distances à l'aide d'un serveur d'application.

extensions peuvent contenir des informations concernant l'authentification, la gestion des transactions, le payement, etc. [2]

Le corps (Body) : c'est le bloc qui contient le corps de message, il offre un mécanisme simple d'échange des informations mandataires destinées au receveur du message SOAP. Cette partie contient les paramètres fonctionnels tels que le nom de l'opération à invoquer, les paramètres d'entrés et de sortis ou des rapports d'erreur [2].

Fault : sert à reporter des erreurs lors du traitement du message, ou lors de son transport. Il ne peut apparaître qu'une seule fois par message. Sa présence n'est pas obligatoire [4]. La figure 4 montre la structure d'un message SOAP :

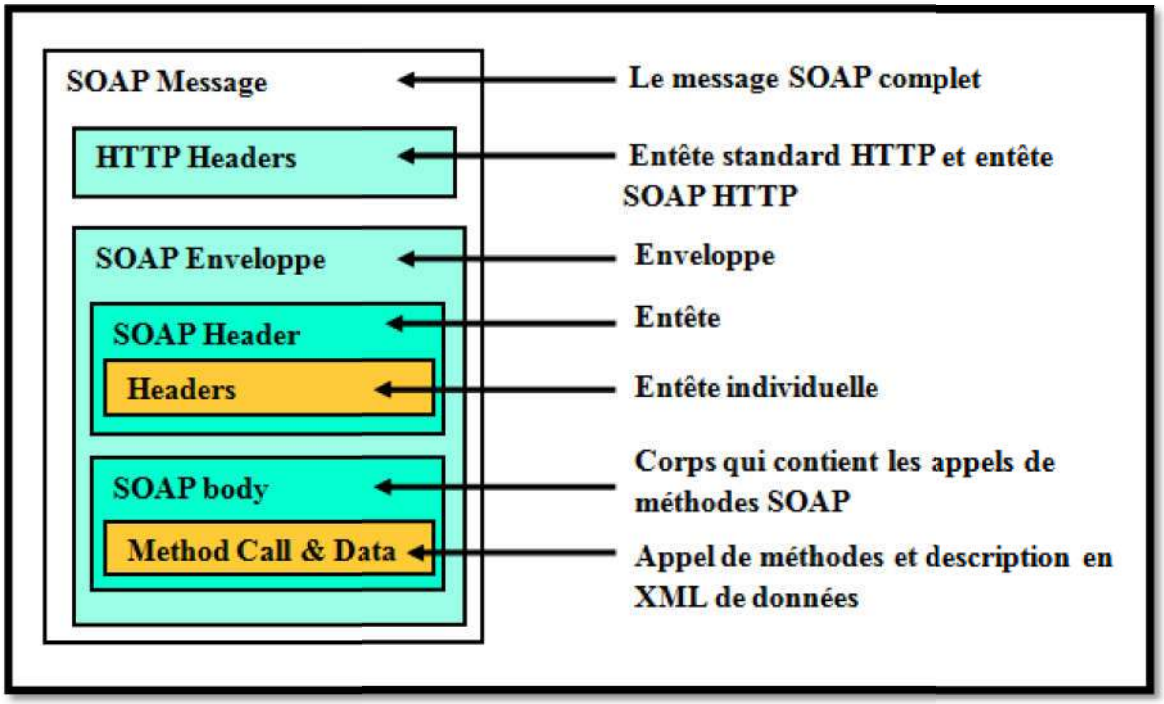

Figure 4 : Structure d'un message SOAP [9]

3.4.3. UDDI (Universal Description, Discovery and Integration) :

UDDI est un protocole basé sur XML qui sert à la publication et à l'exploration d'annuaires de Services Web. Il permet de situer un service Web, de le décrire, et de l'intégrer [4].

UDDI détermine comment nous devons organiser l'information concernant une entreprise et le service web qu'elle offre à la communauté, d'y avoir accès. Nous pouvons donc diviser le concept UDDI en deux portions soit, l'enregistrement de l'information et la découverte de cette information.

Pour les deux portions précédentes, les entreprises et individus utiliseront des API[4] SOAP ou encore l'interface utilisateur fournit par le propriétaire de registre UDDI en question. Ils devront publier séparément des descriptions WSDL de leurs services pour fins d'enregistrement, de ceux de découverte de services web [36].

En effet UDDI fournit des fichiers WSDL différents pour l'enregistrement et la découverte des services web. Tel que vous pouvez le remarquez dans la figure 5 :

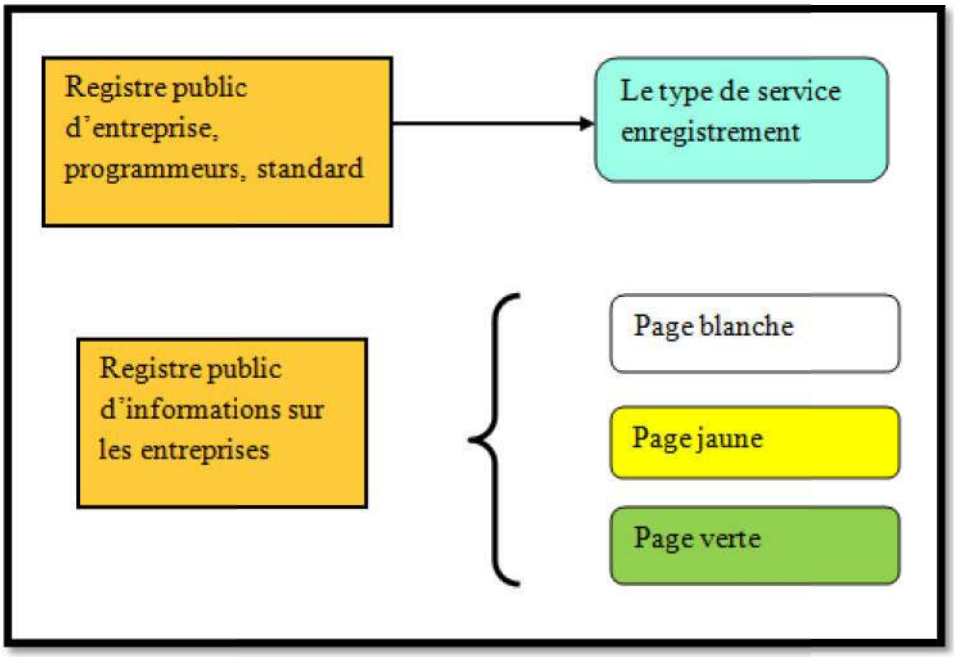

Figure 5 : Données de registre UDDI [9]

Les types d'informations XML que l'on retrouve dans l'enregistrement UDDI sont divisés en trois catégories [3] :

- ✓ Catégories contiennent les informations d'affaire nominative : les pages blanches.
- ✓ Des codes, index descriptif et des diverses catégorisations des services : les pages jaunes.
- ✓ Des règles techniques et technologiques d'interactions : les pages vertes.

3.5. Composition des services web :

Afin de réaliser une tache plus complexe qui satisfait les besoins de l'utilisateur, un service web peut être composé avec d'autres services existants, cela implique l'interaction entre plusieurs services pour atteindre un but précis.

[4] Application Programming Interface : ensemble de classes et des méthodes qui sert de façade par laquelle un logiciel offre des services à autres logiciels

La composition des services Web est [19]le processus de construction de nouveaux services Web à valeur ajoutée, à partir de deux ou plusieurs services Web déjà présents et publiés sur le Web. Un service Web est dit composé ou composite lorsque son exécution implique des interactions avec d'autres services Web, et des changements des messages entre eux afin de faire appel à leurs fonctionnalités. La composition de services Web spécifie quels services ont besoin d'être invoqués, dans quel ordre et comment gérer les conditions d'interaction.

3.5.1. Approches de composition des services web :

Il existe principalement deux grandes approches de composition des services web : composition statique et composition dynamique, nous détaillons chaque approche dans ce qui suit.

3.5.1.1. Composition statique :

Dans l'approche statique, les services web à composer sont choisis à l'heure de faire l'architecture et le design, les composants sont choisis et reliés ensemble avant d'être compilés et déployés [24]. BPEL[5] et WS-CDL[6] sont deux exemples de moteurs de composition statique des services web.

La composition statique des services web peut être élaborée de deux manières différentes : L'orchestration et la chorégraphie.

L'Orchestration : l'orchestration de services Web consiste en la programmation d'un moteur qui appelle un ensemble de services Web selon un processus prédéfini. Ce moteur définit le processus dans son ensemble et appelle les services Web (tant internes qu'externes à l'organisation) selon l'ordre des tâches d'exécution. La Figure 6 illustre l'exécution du moteur (lui-même un service Web) permise par l'enchaînement de l'exécution de deux autres services Web [19].

[5]Business Process Execution Language
[6] Web Service Choreography Description Language

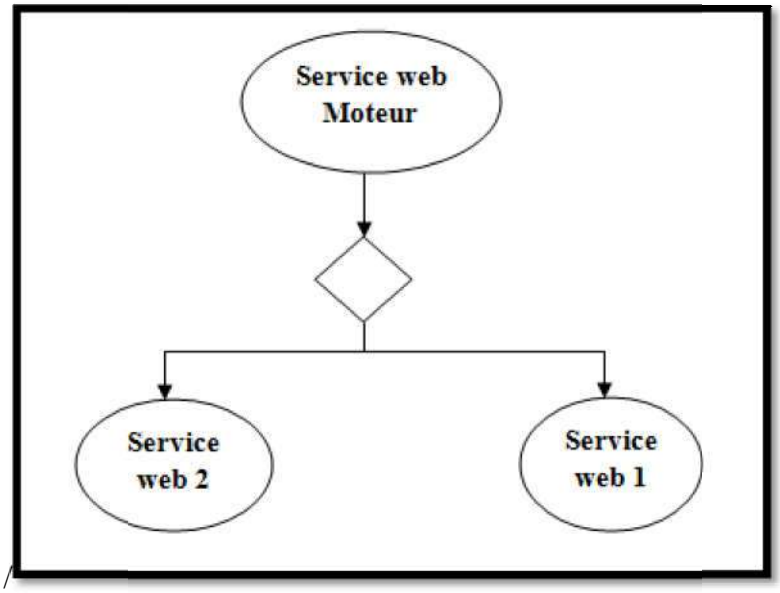

Figure 6: Illustration de l'orchestration [19]

Cet enchaînement est possible via un opérateur d'ordonnancement (représenté par le losange dans la figure). L'exécution de la composition repose sur l'appel du Service Web 1, puis sur l'appel du Service Web 2, réalisés tous deux par le Service Web Moteur [19].

La Chorégraphie : la chorégraphie permet de décrire la composition comme un moyen d'atteindre un but commun en utilisant un ensemble de services Web. La collaboration entre chaque service Web de la collection (faisant partie de la composition) est décrite par des flots de contrôle. Pour concevoir une chorégraphie, les interactions entre les différents services doivent être décrites. La logique de contrôle est supervisée par chacun des services intervenant dans la composition. L'exécution du processus est alors distribuée [19].

Un protocole d'initiation de collaboration entre deux services dans le cadre d'une chorégraphie est présenté dans la figure suivante :

Figure 7 : Illustration de la Chorégraphie [19].

Dans cet exemple, le Service Web 1 demande l'exécution d'une méthode du Service Web 2 par l'intermédiaire d'un envoi de message (Requête de service). Cette requête est acceptée par le service Web 2. Ce dernier envoi un message d'acceptation au Service Web 1(Acceptation). Le Service Web 1 accepte le service proposé par le Service Web 2 en lui envoyant un message (Service admis) accordé (Acceptation) par ce second service. Une fois

ces messages échangés le Service Web 1 peut invoquer le Service Web 2 dans le cadre de la composition [19].

3.5.1.2. Composition dynamique :

Contrairement à la composition statique des services web où le nombre de services fournis est limité et les services à composer sont spécifiés au préalable, l'approche dynamique compte sur l'idée ou les services sont sélectionnés et composés à la volée en fonction des besoins formulés par l'utilisateur [19]. L'approche dynamique répond à des besoins de flexibilité et d'adaptation. Elle permet en effet de générer des plans de compositions adaptés à chaque requête à partir des services disponibles au moment de la composition.

3.6. BPEL (Business Process Execution Language):

BPEL est un langage de composition statique des services Web proposé par BEA, IBM et Microsoft, ce langage est conçu pour supporter les processus métier à travers les services Web. Il est issu de la fusion de deux langages [19]: WSFL[7] d'IBM et XLANG de Microsoft. BPEL4WS combine les caractéristiques d'un langage de processus structuré par bloc (XLANG) avec ceux d'un langage de processus basé sur les processus métier (WSFL).

BPEL supporte deux différentes méthodes pour la description des processus : processus abstrait et processus exécutable.

- **Processus abstrait :** permet de spécifier les messages échangés entre les différents services web participants sans spécifier le comportement interne de chacun d'eux. Ce processus abstrait peut être relié à une composition de type chorégraphie. Les services Web communiquent alors à l'aide d'échanges de messages [19].
- **Processus exécutable** : sert à spécifier l'ordre d'exécution des activités, le partenaire concerné, les messages échangés entre ces partenaires, et les mécanismes 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[19].

Un processus BPEL définit l'ordre dans lequel les services Web impliqués sont composés, soit en séquence, soit en parallèle. BPEL permet de décrire les activités conditionnelles. Une invocation d'un service Web peut par exemple s'appuyer sur le résultat de l'invocation d'un autre service Web. Avec BPEL, il est possible de créer des boucles, de déclarer des variables, de copier et d'attribuer des valeurs ainsi que d'utiliser des gestionnaires

[7] Web Service Flow Language

de fautes. Des processus métier complexes peuvent être construits algorithmiquement en utilisant toutes ces constructions.

3.6.1. Eléments d'un processus BPEL :

Les principaux éléments d'un processus BPEL sont : les liens de partenaires, les activités et les données.

- **Les liens de partenaires :** Les liens partenaires (*PartnerLink*) correspondent aux différents services Web invoqués dans le processus. Ils ont chacun un rôle spécifique dans un processus donné. Chaque lien de partenaire est décrit par son nom, son rôle (en tant que service indépendant), et son rôle dans le processus [21].
- **Les activités :** Le procédé dans BPEL est constitué d'activités liées par un flot de contrôle. Ces activités peuvent être basiques ou structurées [21] :

Activités basiques :

- <invoke> : pour invoquer une opération dans un service web.
- <receive> : attendre qu'un client ou un service invoque le processus par envoie de requête.
- <reply> : générer une réponse synchrone.
- <assign> : permet de manipuler les données des variables.
- <throw> : indique les erreurs et les exceptions.
- <wait> : attendre un certain temps.

Activités structurées : la combinaison des activités basiques décrit au dessus forme des activités structurées afin définir un algorithme complexe spécifiant les étapes par lesquelles passe le processus. Parmi ces activités on cite :

- <sequence> : définit un groupe d'activités qui vont invoquées par l'ordre d'apparition dans la séquence.
- <flow> : définit un groupe d'activités qui vont invoquées en parallèle.
- <switch> : pour l'acheminement conditionnel.
- <while> : pour les boucles.

:

- **Les données :** Le processus 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 processus. Elles sont utilisées dans les expressions et les opérations d'affectation [21].

3.6.2 Caractéristiques de BPEL :

Certaines des caractéristiques saillantes de BPEL sont [20] :

- ✓ Sémantique de séquençage riche incluant le traitement parallèle et asynchrone.
- ✓ Fournit des capacités de gestion des pannes riches
- ✓ Fournit des fonctionnalités asynchrones de gestion des événements asynchrones riches qui permettent des alertes basées sur le temps ainsi que des événements hors bande tels que l'annulation d'ordre
- ✓ Utilise les services Web comme modèle pour la décomposition et l'assemblage des processus, c'est-à-dire que chaque processus BPEL est un service Web et peut être composé comme tel dans d'autres processus BPEL
- ✓ Utilise XML et XPath pour l'accès aux données et la manipulation.

3.7. Les Avantages des Services Web :

Les Services web deviennent de plus en plus populaires et utilisées fréquemment grâce à ses avantages intéressants, parmi lesquels on cite :

La Simplicité : Les services web réduisent la complexité des branchements entre les participants. Cela se fait en ne créant la fonctionnalité qu'une seule fois plutôt qu'en obligeant tous les fournisseurs à reproduire la même fonctionnalité à chacun des clients selon le protocole de communication supporté [2].

Ouverture : les Services Web permettent de tirer une valeur économique supplémentaire des infrastructures informatiques existantes et des plates-formes ouvertes tel que l'Internet [36].

Composante logicielle légèrement couplée : L'architecture modulaire des services web, combinée au faible couplage des interfaces associées, permet l'utilisation et la réutilisation de services qui peuvent facilement être recombinés à différents autres applications [2].

L'hétérogénéité : les Services Web tiennent compte à la diversité des plates-formes et des applications qui existent à l'intérieur et l'extérieur de l'entreprise. Ils pourront créer plus de valeurs d'affaires en tirant partie de cette réalité inter et extra entreprise, en permettant a diverses ressources informatiques et d'affaires de se brancher et de communiquer entre elles [36].

L'interopérabilité : C'est la capacité des services web d'interagir avec d'autres composantes logicielles via des éléments XML et utilisant des protocoles de l'Internet [2].

Auto-descriptivité : Les services web ont la particularité d'être auto-descriptifs, c'est à dire capables de fournir des informations permettant de comprendre comment les manipuler. La capacité des services à se décrire par eux-mêmes permet d'envisager l'automatisation de l'intégration de services [2].

4. Qualité de service dans les Services Web :

Afin de choisir le meilleur service en termes des exigences des clients, nous avons besoin d'une technique permettant d'évaluer et de comparer les différents services offrant les mêmes fonctionnalités. Dans ce contexte, la qualité du service peut être définie comme [6] la différenciation des services de placement en fonction des exigences des clients et des applications. Cela implique l'évaluation des propriétés non-fonctionnelles du service décrit dans un modèle de qualité.

4.1. Définition :

La qualité de service (en anglais QoS : Quality of Service) désigne [5] la capacité d'un service à répondre par ses caractéristiques aux différentes exigences de ses utilisateurs en termes par exemple de disponibilité (ex. taux de rejet), fiabilité (ex. temps moyen entre deux pannes), performance (ex. temps de réponse) et coût (ex. économique, énergétique). Les principales composantes de la qualité de service seront fournies par des métriques caractérisées par un type, une unité et une fonction de calcul.

4.2. QdS spécifiques et génériques :

Les attributs de QdS peuvent être classifiés en 2 parties : les QdS spécifiques, et les QdS génériques :

4.2.1 QdS spécifiques :

Les QdS spécifiques sont des qualités qui concernent une application particulière, et qui sont en relation avec sa logique métier. Il s'agit des événements à mesurer afin de diagnostiquer des éventuels problèmes du système [2].

4.2.2 QdS génériques :

Les QdS génériques peuvent être divisées en deux parties : des paramètres mesurables et paramètres non mesurables :

- ✓ *Attributs mesurables :* La disponibilité, La performance, la fiabilité, la Scalabilité, la robustesse.
- ✓ *Attributs non mesurables :* La sécurité, la réputation, le prix d'exécution.

4.3. Exigences de QdS dans les Web services :

Les exigences principales pour supporter la QdS dans les services Web sont représentées dans le tableau suivant :

Attribut	Définition	Type
Disponibilité	L'absence d'interruptions de service. elle représente la probabilité qu'un service est disponible. Des valeurs plus élevées signifient que le service est toujours prêt à l'emploi tandis que les valeurs plus petites indiquent imprévisibilité quant à savoir si le service sera disponible à un moment donné [7].	*Mesurable*
Accessibilité	Représente le degré avec lequel une demande de service Web est servie. Un degré élevé d'accessibilité signifie qu'un service est disponible pour un grand nombre de clients et que les clients peuvent utiliser le service relativement et facilement [7].	*Mesurable*
Performance	Mesurée en termes de deux facteurs: le débit et la latence. Le débit représente le nombre de demandes de service Web à une période de temps donnée. Temps d'attente représente la durée entre l'envoi d'une requête et la réception de la réponse. Un débit plus élevé et les valeurs de latence inférieures représentent une bonne performance d'un service Web [7].	*Mesurable*
Fiabilité	Représente la capacité d'un service à fonctionner correctement et de manière cohérente et de fournir la même qualité de service en dépit des défaillances du système ou réseau. La fiabilité d'un service Web est généralement exprimée en termes de nombre d'échecs de transactions par mois ou année [7].	*Mesurable*
Scalabilité	Se réfère à la capacité à servir des demandes toujours, malgré les variations du volume des demandes. Haute accessibilité des services Web peut être atteint par la construction de systèmes hautement évolutifs [7].	*Mesurable*
Intégrité	L'aspect de la qualité de la manière dont le service web maintient l'exactitude de l'interaction par rapport à la source. La bonne exécution des transactions de services Web fournira l'exactitude de l'interaction. Une transaction se réfère à une séquence d'activités à traiter comme une seule unité de travail. Toutes les activités doivent être terminées pour effectuer la transaction réussie. Lorsqu'une transaction ne se termine pas, toutes les modifications apportées sont annulées [8].	*Mesurable*
Sécurité	Concerne des aspects tels que la confidentialité, la non-répudiation en authentifiant les parties concernées, le cryptage des messages, et en fournissant un contrôle d'accès. La sécurité a une importance supplémentaire parce que l'invocation de service web se produit sur l'Internet public. le fournisseur de services peut avoir différentes approches et niveaux d'assurer la sécurité en fonction du demandeur de service [8].	*Non mesurable*
Robustesse	Établit la capacité du système pour donner une réponse précise en présence d'entrées non valides, incomplètes ou contradictoires. Ceux-ci peuvent être estimés à côté du fournisseur ou testés et faisant une moyenne des dernières expériences [6].	*Mesurable*
Réputation	Mesure des clients de l'expérience qu'ils avaient à utiliser le service. La valeur de la réputation est la moyenne de la réputation de ce service partagé par tous les clients, à savoir, somme des réputations, divisé par le nombre de clients. Normalement, les réputations sont définies dans une plage de valeurs [7].	*Non mesurable*
Prix d'exécution	Montant d'argent que le client doit payer pour l'exécution de la demande du fournisseur de services. Le fournisseur peut proposer un prix fixe ou un prix en fonction des valeurs des propriétés non-fonctionnelles proposées par le service [6].	*Non mesurable*

Tableau 1 : Définitions des exigences de QdS pertinentes.

5. Conclusion :

En résumé, on peut dire que les services web représentent une technologie essentielle pour l'implémentation des applications distribuées. Ils sont caractérisés par la réutilisabilité, l'interopérabilité, et leurs indépendances aux plates-formes et aux systèmes d'exploitation. Nous avons présenté dans ce chapitre les notions et les concepts de base concernant les Web services et l'AOS, nous avons aussi examiné les concepts connexes de QdS pour les Web services. Le chapitre suivant présentera les différentes approches de Monitoring de QdS.

Chapitre 2 : Les techniques de Monitoring des QdS

1. Introduction :

Le Monitoring de QdS est une étape nécessaire pour vérifier les performances des services réels .Nous présentons dans ce chapitre la notion de monitoring de QdS et un bref état de l'art sur les différentes techniques de Monitoring des QdS des services Web en donnant les avantages et les inconvénients de chaque technique afin de bien situer l'approche proposée, et nous conclurons notre chapitre par une présentation de la programmation orientée aspect.

2. Définition de Monitoring :

Le Monitoring est une activité de surveillance qui permet de mesurer et surveiller une tâche précise. Alors on peut dire que le Monitoring des qualités de service web est une activité qui consiste à surveiller et mesurer les valeurs de QdS afin de vérifier la conformité de ces métriques avec les valeurs publiées par le fournisseur de service.

3. L'objectif de Monitoring :

Dans les services Web, des outils de surveillance sont utilisés pour [11]:

- ✓ Améliorer le processus de sélection de service Web et la découverte, ce qui permet à la qualité de service sur la base des recherches parmi des services fonctionnellement similaires.
- ✓ Technique d'auto-réparation (en anglais : Self-Healing), comme l'adaptation dynamique et la récupération dynamique appliquée à certains attributs de qualité (disponibilité, d'évolutivité, de capacité et de fiabilité) lorsque certains d'entre eux pas atteint le niveau désiré.
- ✓ Détecter les violations de SLA, les métriques de qualité, basés sur SLA[8] sont utilisés pour évaluer et contrôler le service Web.

4. Les outils de Monitoring des services web :

Les outils de monitoring sont des systèmes qui captent, collectent, filtrent, et analysent les informations à partir d'un système logiciel lors de l'exécution. Ils peuvent être utilisés selon [11] deux stratégies :

[8] Service Level Agreement : un contrat qui définit la QdS requise entre un fournisseur de service et un client.

- **Surveillance passive :** le moniteur est un renifleur (en anglais. Sniffer), intercepte les messages échangés entre le fournisseur et le consommateur de services, dans le but d'obtenir la qualité de service. Dans cette stratégie, une interaction directe avec le fournisseur ou le consommateur est réduite au minimum.
- **Surveillance active :** le moniteur a envoyé des demandes de service directement au fournisseur de services, agissant en tant que consommateur.

Les systèmes de Monitoring peuvent être configurés différemment selon trois composantes, comme montre la figure suivante :

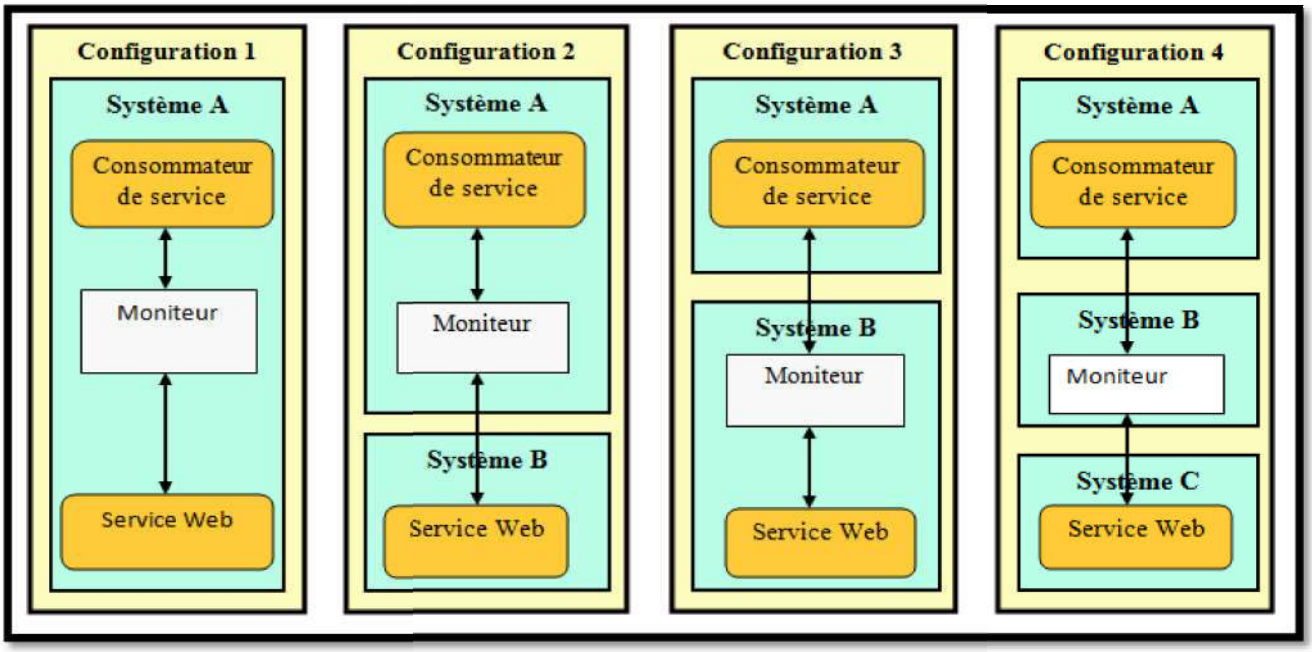

Figure 8 : Les configurations d'un moniteur [11]

- ✓ **Configuration 1 :** Le consommateur de services, le moniteur et le service Web sont dans le même système (système A).
- ✓ **Configuration 2 :** Le consommateur de services et le moniteur sont dans le même système (système A), et le service Web est dans un autre système (système B).
- ✓ **Configuration 3** : Le consommateur de services est dans un système (système A), et le moniteur et le service Web sont ensemble dans un autre système (système B).
- ✓ **Configuration 4 :** Le consommateur de service est dans le système A, le moniteur et le service Web sont dans le Système B et Système C, respectivement.

5. Techniques de Monitoring des QdS des services web :

Le Monitoring de QdS est une procédure importante dans un environnement de service Web. Il correspond à une étape d'observation du comportement du service et d'extraction des métriques nécessaires pour effectuer les mesures des QdS. Différentes approches de mesure de QdS ont été proposées, nous détaillons quelques approches dans la section suivante.

5.1. Le Timer inséré dans le code :

Les auteurs de [8] ont proposés une méthode simple pour mesurer les caractéristiques de performance des services Web, l'idée principale de cette méthode est d'ajouter des fonctionnalités supplémentaires dans le proxy de service. Cela se fait selon quatre étapes :

Etape1 : Générer le proxy de service à partir de WSDL :Typiquement, les proxies de service ne sont pas écrites par le programmeur. Ils peuvent être facilement générés à partir du fichier WSDL.

Etape 2 : Modifier le proxy de service généré : La modification se faite en ajoutant quelques lignes de code. Ces lignes ajoutées instancient un objet ***Timer*** pour mesurer le temps qu'il faut pour se lier au serveur et invoquer une méthode.

Etape 3 : Recompiler le proxy de service généré : Le fichier source de proxy de service modifié doit être recompilé, simplement en utilisant la commande *javac* ou en utilisant tout autre compilateur.

Etape 4 : Développer le programme Client : Développer une application cliente, qui peut utiliser le proxy de service pour appeler le service Web. Cela pourrait être un programme Java simple ou un AWT / Swing application basée sur Java GUI. L'insertion de code de Monitoring est illustrée dans la figure 9.

Cette technique de Monitoring est simple mais elle exige l'accès direct au code et l'utilisation du même langage que le client afin d'effectuer la mesure.

```
import java.net.*;
import java.util.*;
import org.apache.soap.*;
import org.apache.soap.encoding.*;
import org.apache.soap.rpc.*;
import org.apache.soap.util.xml.*;
import mytimer.Timer;
public class EchoServiceProxy {
private Call call = new Call ();
private URL url = null;
private String SOAPActionURI = "";
private SOAPMappingRegistry smr = call.getSOAPMappingRegistry ();
public EchoServiceProxy () throws MalformedURLException {
....
// Démarrer le Timer
Timer timer = new Timer ();
timer.start ();
Response resp = call.invoke (url, SOAPActionURI);
// Arrêter le Timer
timer.stop ();
// Afficher le temps de réponse en calculant la différence
System.out.println ("Response Time = " + timer.getDifference());
// Vérifier la réponse
if (resp.generatedFault()) {
Fault fault = resp.getFault (); throw new SOAPException (fault.getFaultCode(),
fault.getFaultString());
} else
{
Parameter retValue = resp.getReturnValue ();
return (java.lang.String) retValue.getValue ();
}
}
```

Figure 9 : Le Timer de Monitoring [8]

5.2. Monitoring de QdS en utilisant la Programmation Orientée Aspect :

Une autre approche de mesure de QdS est basée sur la programmation orientée aspect (en anglais, AOP : Aspect Oriented Programming). C'est un paradigme de programmation [15] qui permet de séparer l'implémentation de toutes les exigences, fonctionnelles ou non, d'un logiciel. Le principe est donc de coder chaque problématique séparément et de définir leurs règles d'intégration pour les combiner en vue de former le système final.

La solution proposée par les auteurs de [22], permet de mesurer le temps de réponse du service web, l'heure de début et l'heure de fin de l'appel de service Web sont acquises et le temps de réponse du service Web est calculé en utilisant la programmation orientée aspect.

L'idée principale de cette méthode est qu'avant que la méthode appelant le service commence l'exécution, un timeAspect pointe vers les codes et enregistre l'heure de début. Après l'exécution de la méthode d'appel de service, le timeAspect tisse également les codes et enregistre l'heure de fin. Le temps de réponse est égal à la soustraction de l'heure de début à l'heure de fin. Les principaux processus de l'outil de mesure proposé sont représentés dans la figure 10:

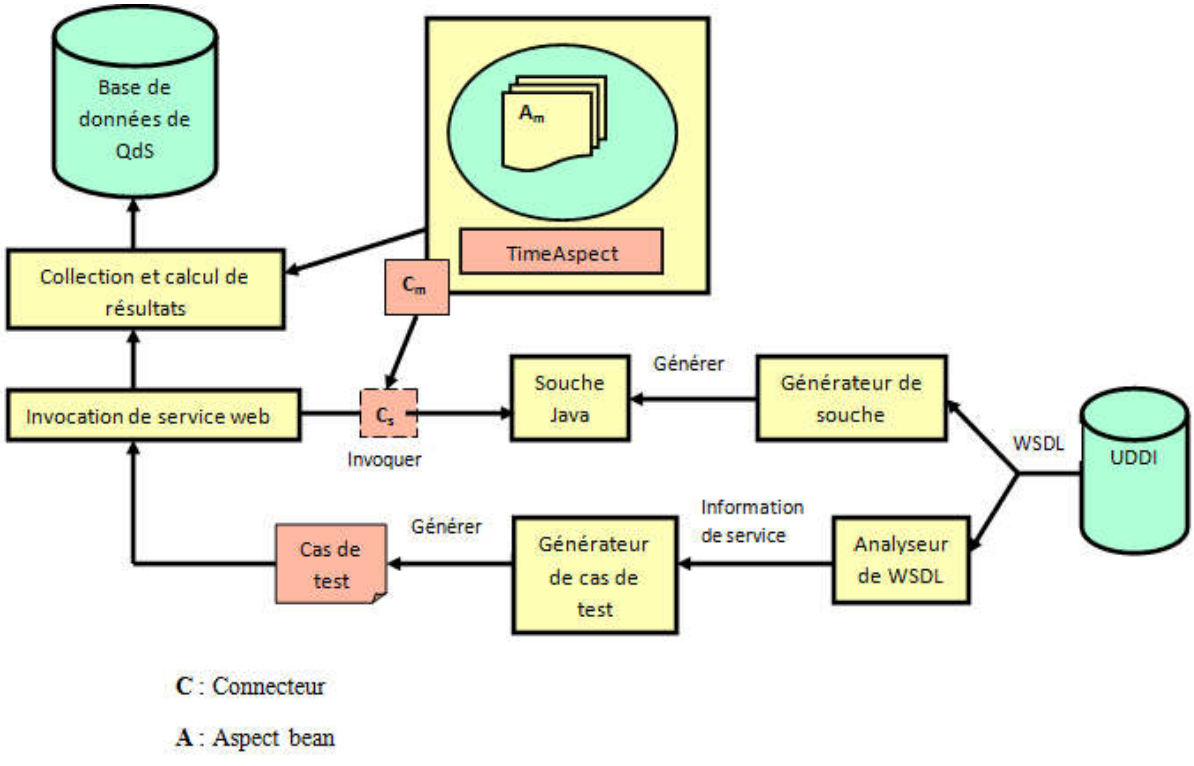

Figure 10 : Monitoring QdS en utilisant L'Aspect.

L'avantage de cette approche réside dans le fait que le monitoring effectue sans avoir besoin d'accéder à l'implémentation du service du côté serveur, mais il reste toujours le problème de la relation directe avec le langage de programmation, L'aspect doit être approprié avec le langage de programmation coté client.

5.3. Modification de la bibliothèque SOAP :

Une approche proposée par l'auteur de [33] pour la mesure automatique des valeurs de QdS pour les services web. L'idée principale de cette approche est de modifier la bibliothèque SOAP pour enregistrer l'information nécessaire pour la mesure de performance.

Pour démontrer la collection des mesures de performance du côté client, un prototype a été implémenté en utilisant la modification du la bibliothèque de gestion des messages SOAP. Le code permettant l'archivage (logging) est ajouté avant que le message SOAP soit envoyé, et au moment de la réception du message de réponse. Ces informations sont envoyées par la suite à une troisième entité responsable du stockage et la mise à jour des mesures.

L'avantage de cette approche réside dans le fait qu'elle n'engendre pas une grande surcharge sur le CPU. En plus, elle n'a pas besoin de configurer le code du client comme pour le cas dans l'approche basée sur le proxy.

La limitation que présente cette approche est la dépendance de l'implémentation et de plateforme. La modification de bibliothèque a besoin d'être disponible sur les différentes implémentations et plateformes. C'est une modification qui doit être établie notamment dans les bibliothèques SOAP des clients ainsi que des fournisseurs.

5.4. Une approche inter-couches pour le monitoring de performance :

Une approche proposée dans [13], pour l'extraction détaillée et en temps réel des informations concernant le comportement du service web et sa performance, en se basant sur l'analyse des protocoles TCP/IP et HTTP. L'idée principale de cette approche est de capturer les paquets composant les messages SOAP entrants et sortants et analyser les données TCP. Cette approche utilise particulièrement les paramètres de la couche transport pour dériver les métriques et les mesures de la performance. Ces paramètres peuvent être la taille de la fenêtre de publicité, le temps d'aller retour d'un message (RTT[9]), numéro de séquence en cours d'utilisation, le Timer, etc.…

L'un des avantages de cette approche réside dans le fait que le programme de Monitoring peut être complètement indépendant du code de client, c.-à-d. il n'exige pas l'accès au code client.

Le problème est de trouver les paquets transmis entre le client et le fournisseur et les itinéraires qu'ils prennent, il peut impliquer un problème de sécurité et les résultats dépendent fortement de l'emplacement réseau. Un autre problème sera posé [2] si les messages SOAP sont chiffrés ou comprimés, le programme de monitoring sera inefficace puisqu'il doit

[9] Round Trip Time : le temps que met un signal pour parcourir l'ensemble d'un circuit fermé

déchiffrer et décomprimer. De grandes ressources de CPU seront alors nécessaires, et la complexité du programme augmente considérablement.

5.5. Approche de proxy :

C'est un composant introduit entre le client et le fournisseur lors de l'appel de service. Les proxies de QdS sont des parties intermédiaires qui router le trafic et mesurer les valeurs de propriétés non-fonctionnelles, Ils contiennent de véritables mesures clients, réponse réelle au scénario réel [6]. Cette approche [2] consiste à utiliser le proxy comme médiateur de communication entre le client et le serveur. Les messages échangés entre le client et le serveur seront donc visibles par le proxy, et les attributs de performance seront mesurés par ce proxy.

Les avantages majeurs d'une telle approche est que le programme de monitoring peut être mis en œuvre indépendamment du matériel et de la plateforme, et il entraîne moins de surcharge de CPU en comparant avec l'approche de monitoring de paquets de bas niveau [2].

Les inconvénients d'une telle approche sont dus au fait que le code client à besoin d'être configuré pour utiliser le proxy et les résultats dépendent de l'emplacement du proxy. De plus, elle ne peut pas résoudre le problème de transformation direct des messages SOAP [2].

6. Synthèse des travaux:

Comme nous avons indiqué ci-dessous, il existe de nombreuses approches de monitoring de QdS pour les services web. Chacun d'entre eux a ses avantages et ses inconvénients. Dans cette section, nous faisons une synthèse des travaux précédents tout en comparant ces travaux avec notre approche. La synthèse est présentée dans le tableau suivant :

Approches	QdS mesurés	Avantages	Inconvénients
Approche de Timer	Temps de réponse	Méthode simple a réalisée	Exige l'accès direct au code
L'utilisation de l'Aspect	Temps de réponse + la disponibilité	Séparation de la mesure du code du client	Dépendance de l'implémentation
Modification de la bibliothèque SOAP	Temps de réponse + débit	Mesure automatique	Dépendance de l'implémentation
Approche inter-couches	Temps de réponse + débit	Pas de modification du code du client et du service.	Grande charge du CPU
Approche de proxy	Temps de réponse + débit	Indépendance du matériel et de la plateforme	Résultats dépendent de l'emplacement du proxy
Notre approche	Temps d'exécution + le temps de réponse + la disponibilité	Mesure d'un seul service web + calcul des métriques d'un ensemble des services web composés	Dépendance de langage de programmation

Tableau 2: Synthèse des approches.

7. Présentation de la programmation orientée aspect (AOP) :

Dans la section précédente, nous avons présenté les différentes techniques de monitoring des QdSs existantes. Dans notre travail, nous nous intéressons à la technique de la programmation orientée aspect, nous présentons ce paradigme de programmation dans la section suivante.

La programmation orientée aspect (Aspect Oriented Programming) est [15] un paradigme de programmation qui permet de traiter séparément les préoccupations transversales, qui relèvent souvent des aspects techniques (Journalisation, Sécurité, Transaction..) et des préoccupations métiers, qui constituent le cœur d'une application.

Cette nouvelle méthode de programmation permet d'implémenter chaque problématique indépendamment des autres, puis, de les assembler selon des règles bien définies. La programmation orientée aspect promet donc une meilleure productivité, une meilleure réutilisation du code et une meilleure adaptation du code aux changements.

7.1. Aspect :

Un aspect est une entité logicielle qui capture une fonctionnalité transversale à une application [23]. Trois éléments principaux dans un aspect : les coupes (pointcuts), les codes advice et le mécanisme d'introduction. Les coupes définissent où l'aspect doit être intégré dans une application et les codes advice définissent ce que fait l'aspect (le quoi). Le mécanisme d'introduction permet d'ajouter du contenu structurel dans une application.

7.2. Point de jonction :

Un point de jonction (en anglais, jointpoint) est un point dans le flot de contrôle d'un programme dans lequel un ou plusieurs aspects peuvent être appliqués [23]. Ils peuvent être de différents types, c'est-à-dire qu'ils font références à des événements du programme : appel de méthode, appels de constructeur, les attributs et levés des exceptions.

7.3. Point de coupure :

Un point de coupure (en anglais, pointCut) désigne un ensemble de point de jonctions, il existe plusieurs types de point de coupures :

- Les coupes d'appels de méthodes : désignant un ensemble d'appels de méthodes.
- Les coupes d'exécution de méthodes : désignant l'exécution d'une méthode.
- Les coupes de modification de données : désignant des instructions d'écritures sur un ensemble d'attributs.

7.4. Code advice :

Les codes advice (Advice code) sont des blocs de code qu'exécutera un aspect, ils caractérisent le comportement de l'aspect, chaque code advice d'un aspect doit être associé à une coupe pour être exécuté. En effet, un code advice [23]n'est jamais appelé manuellement, mais il est invoqué chaque fois qu'un point de jonction, sélectionné par la coupe à la quelle il est associé.

7.5. Mécanisme d'introduction :

Le mécanisme d'introduction est un mécanisme d'extensions permettent d'introduire de nouveaux éléments structuraux aux code d'une application [23], il permet d'étendre la structure de l'application et non pas le comportement de cette dernière.

Le mécanisme d'introduction ne s'appuie pas sur la notion de coupe mais va opérer sur des emplacements bien définis dans le programme. On peut dire que le mécanisme d'introduction est pour l'orientée aspect ce que l'héritage est pour l'orientée objet puisque ces deux derniers permettent d'étendre la structure et non pas le comportement de l'application.

7.6. Tissage :

Une application orientée aspect contient des classes et des aspects, l'opération qui prends en entrée les classes et les aspects et produit une application qui intègre les fonctionnalités des classes et des aspects est appelée le tissage d'aspect (en anglais, aspect weaving), et le programme qui réalise cette opération est appelé tisseur d'aspects (aspect weaver).

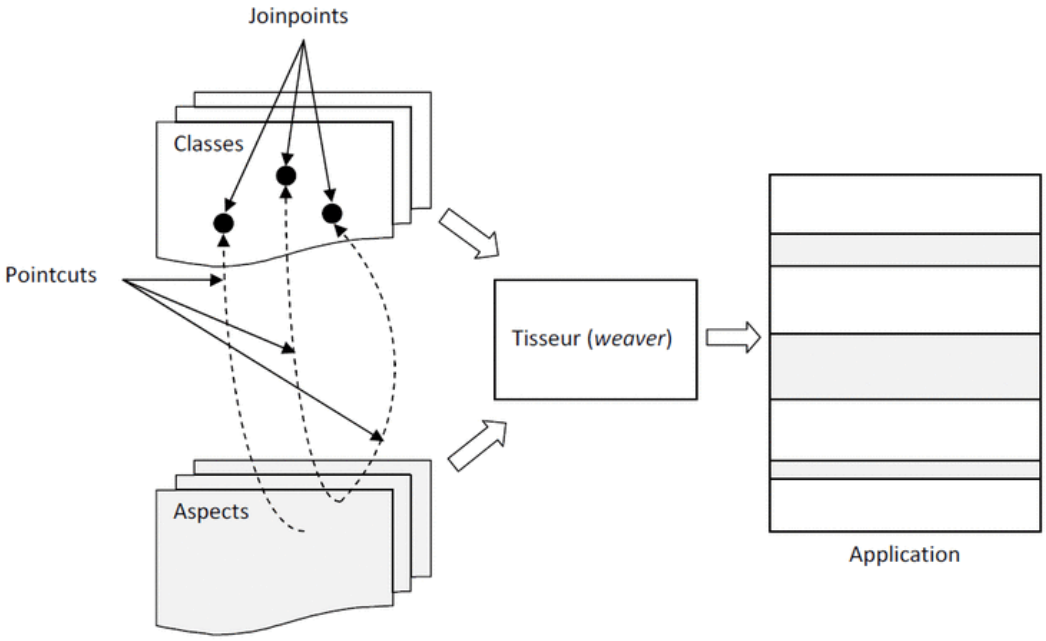

Figure 11 : Tissage des aspects [23]

8. Conclusion :

Différentes approches de mesure de QdS des services web ont été proposées, nous avons présentés dans ce chapitre un aperçu sur quelques ces approches, nous avons montré les avantages et les inconvénients de chaque technique, et nous avons conclu notre chapitre par une synthèse des travaux de recherche tout en comparant ces travaux avec l'approche que nous avons proposés. Dans le prochain chapitre nous allons introduire notre approche et les mesures de QdS proposés pour des le monitoring des services web composés.

Chapitre 3 : l'approche proposée

1. Introduction :

Dans les deux chapitres précédents, nous avons présenté un bref état de l'art sur les services web et la qualité de service ainsi que les différentes approches de monitoring de QdS web existantes actuellement. Dans ce chapitre nous introduisons l'approche que nous avons proposée pour le monitoring de QdS d'une composition des services web, nous commençons d'abord par une représentation de l'approche globale, puis nous détaillons les différents modèles de composition des services web, ensuite nous montrons les mesures de base pour chaque modèle de composition, et nous finissons par une représentation de la technologie utilisée pour le monitoring.

2. Aperçu sur l'approche globale :

Les travaux sur les différentes techniques de monitoring de QdS présentées dans le chapitre 2 permettent la surveillance de QdS d'un seul service web, notre objectif est de fournir une solution au problème de surveillance de QdS d'un ensemble de services web composés en suivant des processus métiers distribués mis en œuvre sur BPEL. Pour bénéficier les avantages de l'utilisation de la programmation orientée aspect, nous proposons d'utiliser la technologie Aspect pour le mécanisme de monitoring.

Notre travail consiste à mesurer séparément la QdS de chaque service web participé dans le processus, puis collecter les mesures de toutes les QdS des services web composés pour obtenir les métriques de monitoring globales.

Comme nous avons indiqué précédemment, un moteur d'orchestration BPEL compose des services Web pour obtenir un résultat spécifique, lorsque le processus métier reçoit une demande du client, il invoque les services web impliqués et répond à l'appelant d'origine (voir figure 12).

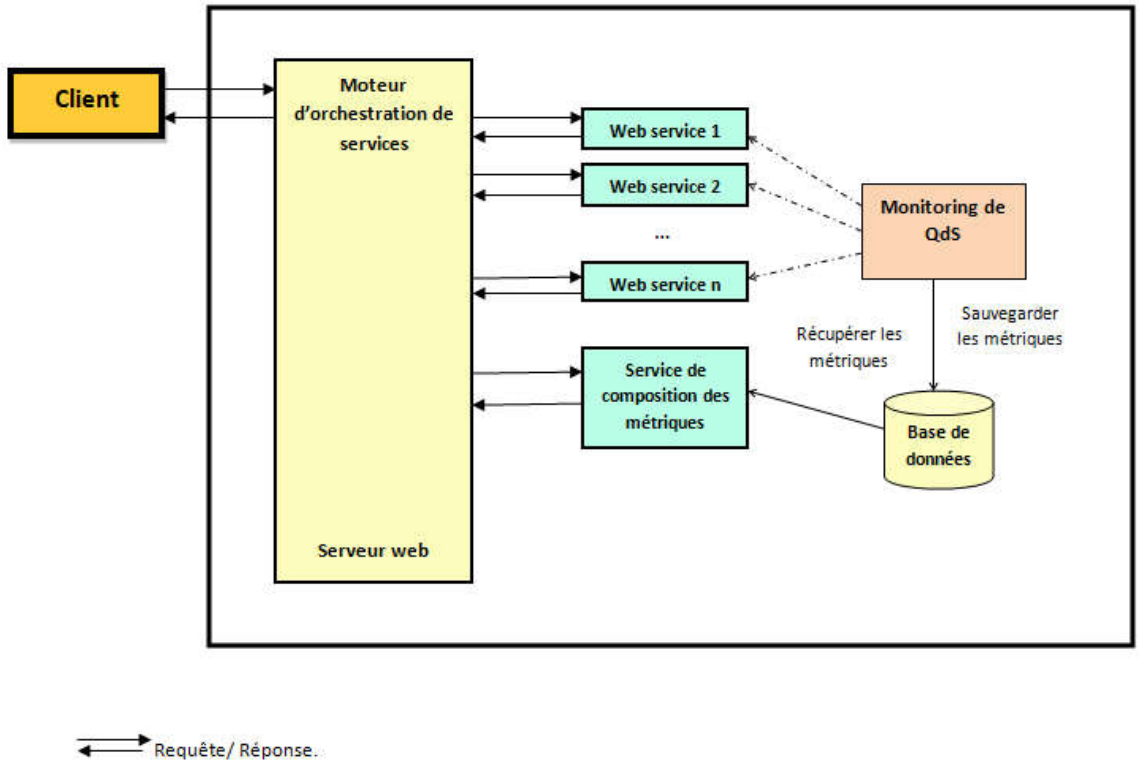

Figure 12 : Architecture globale du système

La phase de monitoring correspond à l'observation et le stockage des paramètres de QdS des services web à orchestrer, elle permet l'interception de ces valeurs lors de l'exécution de service. Cette phase est implémenté en utilisant des intercepteurs de la programmation orientée aspect.

Un intercepteur est défini comme une méthode qui intercepte l'appel lors de l'invocation de service par le processus, il permet de mesurer le temps d'exécution, le temps de réponse, la disponibilité de chaque service web participé, pour les sauvegarder par la suite dans une base de données.

La composition des métriques se fait par un service web spécifique invoqué par le processus métier. Il permet de récupérer les métriques de la base de données et calculer les valeurs des paramètres de QdS globales selon le modèle de composition des services (parallèle, séquentiel, …etc.) et retourner par la suite le résultat au processus pour répond au requête du client.

3. Les modèles de base de composition des services web :

Afin de répondre à une requête complexe du client, les services web disponibles peuvent être combinés pour créer un nouveau service web composite. Cette combinaison se fait en utilisant différents modèles, ces modèles ont illustrés dans cette sous-section.

3.1. Composition séquentielle :

Dans le modèle séquentiel, les services web sont exécutés l'un après l'autre dans une séquence ordonnée, c.-à-d. qu'ils sont dépendants et l'ordre d'exécution est défini et doit être suivi. Le modèle est montré dans le schéma suivant :

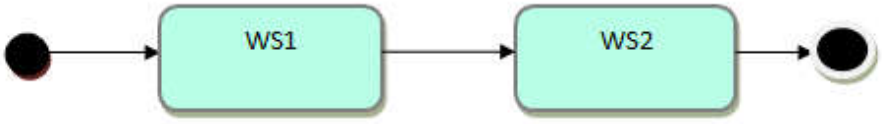

Figure 13 : le modèle séquentiel

Pour la représentation de ce modèle, BPEL défini une activité qui s'appelle « *sequence* ». Ce type d'activité permet de montrer qu'un ensemble d'activités de BPEL seront exécutés séquentiellement.

3.2. Composition parallèle :

La composition parallèle exécute simultanément deux (ou plusieurs) services web, cela signifie qu'ils sont indépendants et l'ordre dans lequel ils sont exécutés n'est pas défini, comme montre le schéma suivant :

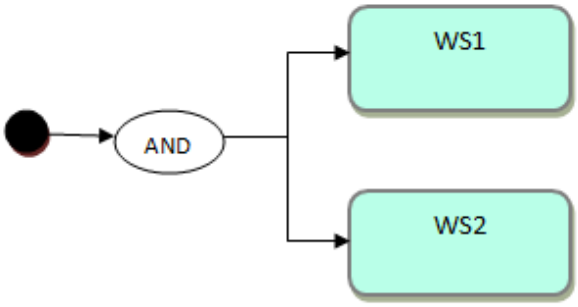

Figure 14 : Le modèle parallèle

L'activité « *flow* » de BPEL, permet de spécifier les activités qui seront exécutées d'une façon concurrente.

3.3. Synchronisation :

Le modèle de synchronisation indique que le processus se poursuivra après l'exécution du modèle parallèle du service Web [7], Il peut être utilisé lorsqu'une certaine décision doit être atteinte après un traitement en parallèle, par exemple choisit le résultat d'un service et rejet l'autre.

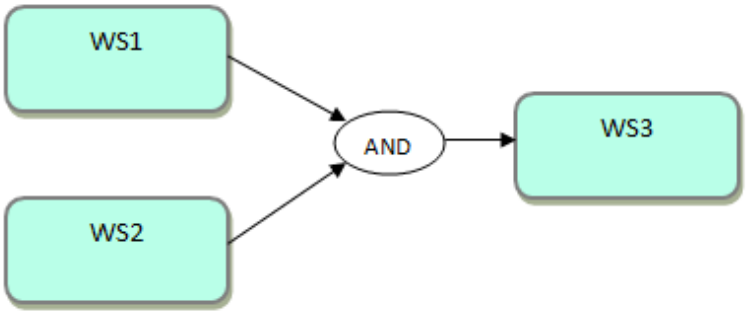

Figure 15 : Le modèle de synchronisation.

BPEL représente ce modèle avec le même type d'activité présenté précédemment (*flow*).L'activité « *flow* »permet de fusionner les liens pour continuer l'exécution dans un seul flux de processus.

3.4. Choix Exclusif :

Le modèle de choix exclusif définit un point dans le flux de travail où un chemin est choisi entre plusieurs, ce choix est fait à l'aide d'une décision prise au moment de l'exécution, cette décision est basée sur une information ou une donnée du processus. Contrairement au *Parallélisme*, un seul lien dans le flux de contrôle est activé [21]. Le modèle du Choix Exclusif est représenté dans la figure 16.

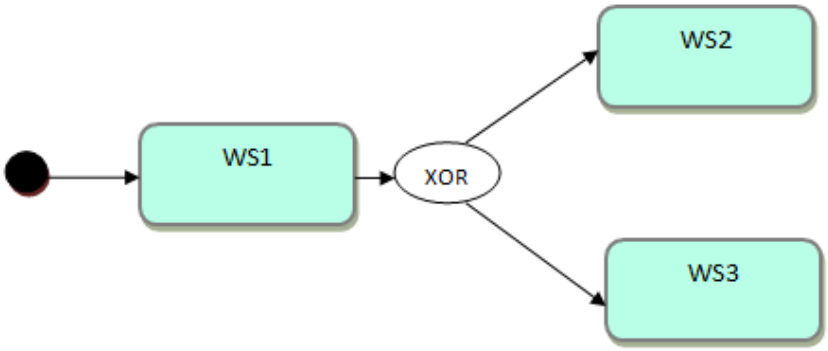

Figure 16 : Le modèle de choix exclusif

BPEL réalise ce type de composition par une activité « *switch* », chaque lien inclus dans le choix exclusif est contenu dans un élément « *case* ». Au moment de l'exécution de « *switch* », chaque « *case* »contient la séquence d'activités qui sera exécutée si la condition liée à« *case* » est vraie.

3.5. Choix différé :

Ce modèle décrit un point dans un processus où certaines informations sont utilisées pour choisir une parmi plusieurs branches alternatives. Ce choix est basé sur une information ou une donnée qui n'est pas nécessairement disponibles au moment où ce point du processus est atteint, la sélection de la branche est alors retardée jusqu'à ce que certains événements arrivent pour donner au procédé l'information requise [21]. Le modèle est illustré dans la figure 17.

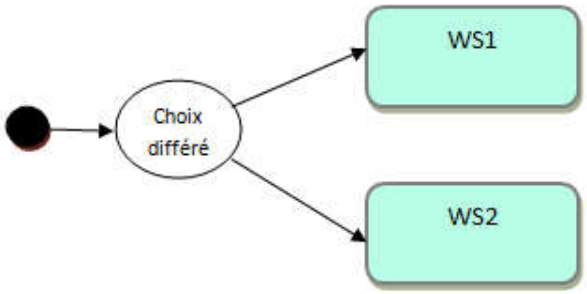

Figure 17 : Modèle de choix différé

La représentation de ce modèle dans BPEL est définie par le type d'activité « *pick* », ce type d'activité attend la réception d'un des messages et continue l'exécution en suivant la branche correspondante au message qui a été reçu. La sélection de la branche est retardée jusqu'à ce que cette activité reçoive un message.

3.6. Choix Multiple (modèle conditionnel):

Le modèle de choix multiple indique que plusieurs branches sont sélectionnées et exécutées en parallèle. Ce choix est fait à l'aide d'une décision prise au moment de l'exécution, cette décision est basée sur une information ou une donnée du processus[21], BPEL utilise le type d'activité « *flow* », l'activité « *empty* »et le concept du « *link* »pour représenter ce modèle. Le schéma suivant montre ce modèle de composition :

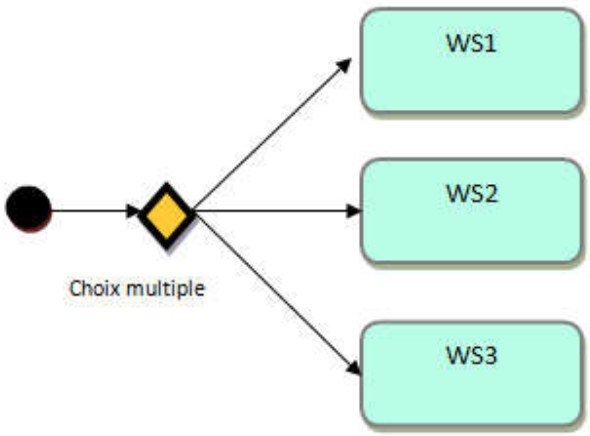

Figure 18 : Le modèle conditionnel.

3.7. Modèle de boucle :

Le modèle de boucle indique qu'un certain service web est exécuté plusieurs fois jusqu'à ce qu'une certaine condition soit satisfaite, la représentation de ce modèle dans BPEL se fait par l'activité « forEach », « RepeatUntil » et l'activité « while ». Nous avons choisi la boucle « while » pour montrer ce modèle.

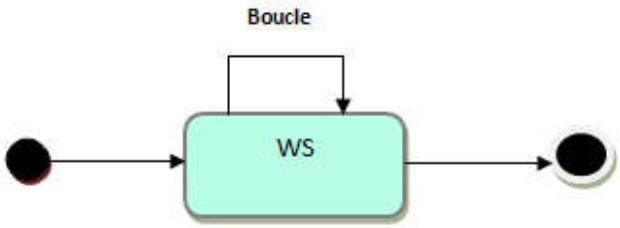

Figure 19 : Le modèle de boucle « while ».

4. Les métriques de Monitoring des services web composés :

Dans la section suivante nous allons présenter les métriques proposées pour le monitoring de QdS des services web pour chaque modèle de composition montré dans la section précédente, les attributs considérés dans notre travail sont : le temps d'exécution, le temps de réponse et la disponibilité, définis respectivement par : TE(s), TR(s), Disp(s). Les services web constituant sont désignés par s_1, s_2, s_3,…, s_n, pour le modèle conditionnel, on dénote l'opération de sélection du service par OS.

La figure 20 illustre l'ensemble des valeurs mesurées pour assurer le monitoring de temps d'exécution, le temps de réponse et le temps de communication ces valeurs sont:

$\mathbf{t_1}$: le temps d'envoi de la requête par le client.

t_2 : le temps de la réception de la requête par le fournisseur de service.

t_3 : le temps d'envoi de la réponse par le fournisseur.

t_4 : le temps de la réception de la réponse par le client.

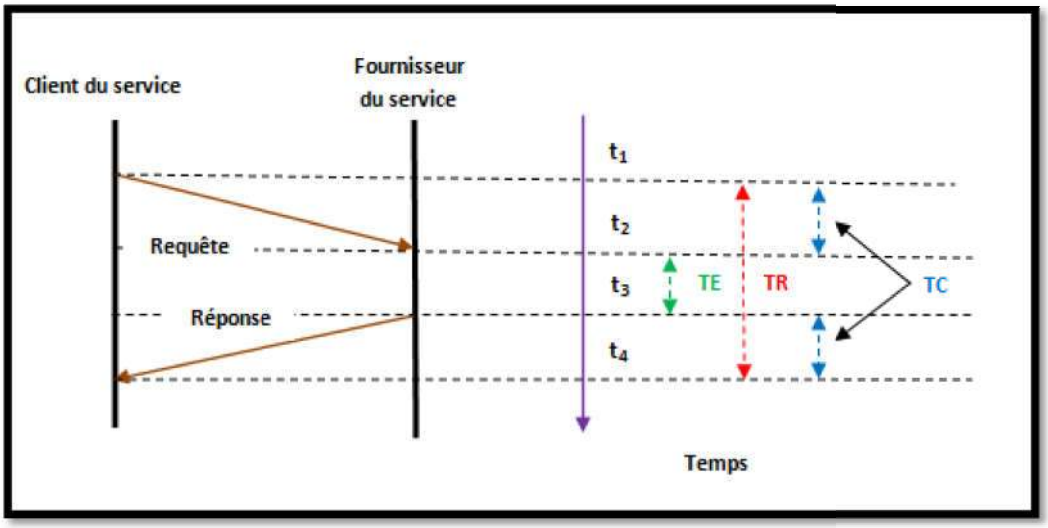

Figure 20 : Schéma illustratif des valeurs de QdS mesurées.

4.1. Mesure de temps d'exécution :

Le temps d'exécution d'un service web est défini par le temps mis par le service pour exécuter la requête [2], mesuré en milliseconde. En suivant le schéma de la figure 20, le temps d'exécution est défini par la formule suivante:

$$TE = t_3 - t_2$$

4.2. Mesure de temps de réponse :

Le temps de réponse d'un service web est défini par le temps que le fournisseur de service a besoin pour répondre à une demande de client [2]. Il peut être divisé en temps de communication et temps d'exécution, mesuré en milliseconde la formule de métrique est :

$$TR = t_4 - t_1$$

4.3 Composition des métriques :

Pour la composition séquentielle des services web, le temps d'exécution et le temps de réponse des services sont défini parla somme des temps d'exécution (temps de réponse) des services web constitutifs, les formules sont présentées comme suit :

Equation 01 : $TE(s)_{séquentiel} = \sum_{i=1}^{n} TE(s_i)$

Equation 02 : $TR(s)_{séquentiel} = \sum_{i=1}^{n} TR(s_i)$

Pour la composition parallèle ainsi que le modèle de synchronisation des services, le temps d'exécution (le temps de réponse) est défini par le temps d'exécution (le temps de réponse) maximal des services web participants, les formules sont présentées comme suit :

Equation 03 : $TE(s)_{parallèle} = \max\{TE(s_i)\}$

Equation 04 : $TE(s)_{synchronisation} = \max\{TE(s_i)\}$

Equation 05 : $TR(s)_{parallèle} = \max\{TR(s_i)\}$

Equation 06 : $TR(s)_{synchronisation} = \max\{TR(s_i)\}$

Pour le modèle de choix exclusif et choix différé, le temps d'exécution et le temps de réponse sont calculés par l'opération de sélection (OS), qui sélectionne l'un des n services Web possibles.

En particulier, le temps d'exécution (temps de réponse) est égal au temps d'exécution (temps de réponse) du service Web sélectionné et est défini comme suit:

Equation 07 : $TE(s)_{choix-exclusif} = \mathrm{TE}(\mathrm{OS}(\mathrm{s_i}))$

Equation 08 : $TR(s)_{choix-exclusif} = \mathrm{TR}(\mathrm{OS}(\mathrm{s_i}))$

Equation 09 : $TE(s)_{choix-différé} = \mathrm{TE}(\mathrm{OS}(\mathrm{s_i}))$

Equation 10 : $TR(s)_{choix-différé} = \mathrm{TR}(\mathrm{OS}(\mathrm{s_i}))$

Pour le modèle conditionnel, le temps d'exécution (et le temps de réponse) globale de la composition est défini par le temps d'exécution (le temps de réponse) maximal des services web sélectionnés, c.-à-d. choisi par l'opération de sélection.

Equation 11 : $TE(s)_{conditionnel} = \max\{TE(OS(s_i))\}$

Equation 12 : $TR(s)_{conditionnel} = \max\{TR(OS(s_i))\}$

Et pour le modèle de boucle, le temps d'exécution (le temps de réponse) est n fois le temps d'exécution (le temps de réponse) de la boucle, ou n est le nombre d'itérations.

Equation 13 : $TE(s)_{boucle} = n \times \mathrm{TE}_{(\mathrm{boucle})}$

Equation 14 : $TR(s)_{boucle} = n \times \mathrm{TR}_{(\mathrm{boucle})}$

4.4. Mesure de disponibilité :

La disponibilité d'un service web est la probabilité que ce service soit opérationnel. Elle peut être calculée de la manière suivante:

Disp = nombre de requêtes réussites / nombre total des requêtes

Pour la composition séquentielle et la composition parallèle, ainsi que le modèle de synchronisation des services web, la disponibilité est définie par le produit des disponibilités de tous les services participants, la formule est présentée comme suit :

Equation 15 : $Disp(s)_{séquentiel} = \prod_{i=1}^{n} Disp(s_i)$

Equation 16 : $Disp(s)_{parallèle} = \prod_{i=1}^{n} Disp(s_i)$

Equation 17 : $Disp(s)_{synchronisation} = \prod_{i=1}^{n} Disp(s_i)$

Pour le modèle de choix exclusif et choix différé, la disponibilité globale est définie par la disponibilité de service web sélectionné par l'opération de sélection, elle est définie comme suit:

Equation 18 : $Disp(s)_{choix-exclusif} = Disp\big(OS(s_i)\big)$

Equation 19 : $Disp(s)_{choix-différé} = Disp\big(OS(s_i)\big)$

Pour le modèle conditionnel, la disponibilité globale est définie par la disponibilité du service web sélectionné.

Equation 20 : $Disp(s)_{conditionnel} = Disp\big(OS(s_i)\big)$

Et pour le modèle de boucle, la disponibilité globale est égale à la disponibilité de la boucle à la puissance n, ou n est le nombre d'itérations.

Equation 21 : $Disp(s)_{boucle} = {Disp_{boucle}}^{n}$

4.5. Synthèse des résultats :

Une synthèse des formules citées dans cette section est présentée dans le tableau 3.

Modèle / Métrique	Temps d'exécution	Temps de réponse	Disponibilité
Séquentiel	$\sum_{i=1}^{n} TE(s_i)$	$\sum_{i=1}^{n} TR(s_i)$	$\prod_{i=1}^{n} Disp(s_i)$
Parallèle	$\max\{TE(s_i)\}$	$\max\{TR(s_i)\}$	$\prod_{i=1}^{n} Disp(s_i)$
Synchronisation	$\max\{TE(s_i)\}$	$\max\{TR(s_i)\}$	$\prod_{i=1}^{n} Disp(s_i)$
Choix-exclusif	$\mathrm{TE}(\mathrm{OS}(s_i))$	$\mathrm{TR}(\mathrm{OS}(s_i))$	$Disp(OS(s_i))$
Choix-différé	$\mathrm{TE}(\mathrm{OS}(s_i))$	$\mathrm{TR}(\mathrm{OS}(s_i))$	$Disp(OS(s_i))$
Conditionnel	$\max\{TE(OS(s_i))\}$	$\max\{TR(OS(s_i))\}$	$Disp(OS(s_i))$
Boucle	$n \times \mathrm{TE}_{(\mathrm{boucle})}$	$n \times \mathrm{TR}_{(\mathrm{boucle})}$	$Disp_{boucle}{}^{n}$

Tableau 3 : Les mesures de QdS des services web composés.

4.6. Exemple illustratif :

Pour montrer les formules précédentes, on prend un exemple de quatre services qui s'exécutent comme suit : le service S1 s'exécute, puis les deux services S2 et S3 s'exécutant en parallèle, ensuite le service S4. L'exemple a illustré dans la figure suivante :

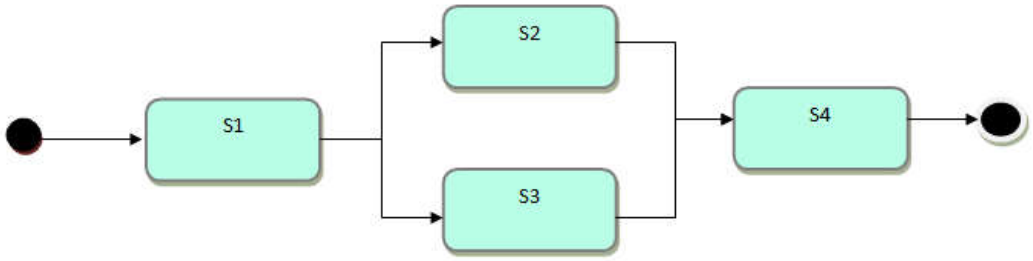

Figure 21 : Exemple de composition des services

Les mesures de monitoring de chaque service sont représentées dans le tableau suivant :

Temps (ms)	t_1	t_2	t_3	t_4	Nombre de requêtes réussites	Nombre total de requêtes	TE = t_3-t_2	TR = t_4- t_1	Disp
S1	1	3	4	5	4	5	1	4	0,8
S2	2	2	5	7	5	5	3	5	1
S3	2	3	4	6	2	3	1	4	0,66
S4	1	5	7	9	8	8	2	8	1

Tableau 4 : Exemple des mesures des services web

Pour obtenir les mesures globales du processus, on applique les formules de composition précédentes comme suit (on note P le processus) :

Le temps d'exécution :

$$TE(P)_{séquentiel} = \sum_{i=1}^{n} TE(s_i) = \mathrm{TE}(S_1) + TE(S_2, S_3)_{parallèle} + \mathrm{TE}(S_4)$$

$$TE(S_2, S_3)_{parallèle} = \max\{\mathrm{TE}(S_2), \mathrm{TE}(S_3)\} = \max\{3,1\} = 3\mathrm{ms}$$

$$TE(P)_{séquentiel} = 1 + 3 + 2 = 6\mathrm{ms}$$

Alors le temps d'exécution globale de processus est $TE(P) = 6\mathrm{ms}$

Pour le temps de réponse :

$$TR(P)_{séquentiel} = \sum_{i=1}^{n} TR(s_i) = \mathrm{TR}(S_1) + TR(S_2, S_3)_{parallèle} + \mathrm{TR}(S_4)$$

$$TR(S_2, S_3)_{parallèle} = \max\{\mathrm{TR}(S_2), \mathrm{TR}(S_3)\} = \max\{5,4\} = 5\mathrm{ms}$$

$$TR(P)_{séquentiel} = 4 + 5 + 8 = 17\mathrm{ms}$$

Alors le temps de réponse globale de processus est $TR(P) = 17\mathrm{ms}$

Pour la disponibilité :

$$Disp(P)_{séquentiel} = \prod_{i=1}^{n} Disp(s_i) = Disp\,(S_1) \times Disp(S_2, S_3)_{parallèle} \times Disp(S_4)$$

$$Disp(S_2, S_3)_{parallèle} = Disp(S_2) \times Disp(S_3) = 1 \times 0{,}66 = 0{,}66 = 66\%$$

$$Disp(P)_{séquentiel} = 0{,}8 \times 0{,}66 \times 1 = 0{,}528 = 52{,}8\%$$

Alors pour cet exemple, la disponibilité globale de processus est $Disp(P) = 52{,}8\%$

5. Méthode de mesure :

La méthode que nous avons utilisés pour mesurer les paramètres de qualité de service est le paradigme orienté aspect qui possède de nombreux avantages parmi lequel une meilleure modularité des systèmes et l'augmentation de la réutilisation du code.

Nous avons utilisé la technique d'interception de la POA pour surveiller les QdS de chaque web service seul, le code advice ici est un intercepteur (en anglais, Interceptors) Ce dernier sera appelé automatiquement avant que l'opération du service web soit exécutée et restera autour lorsque la méthode revient. On montre un exemple de mesure de temps d'exécution d'un service web par un intercepteur dans la figure suivante :

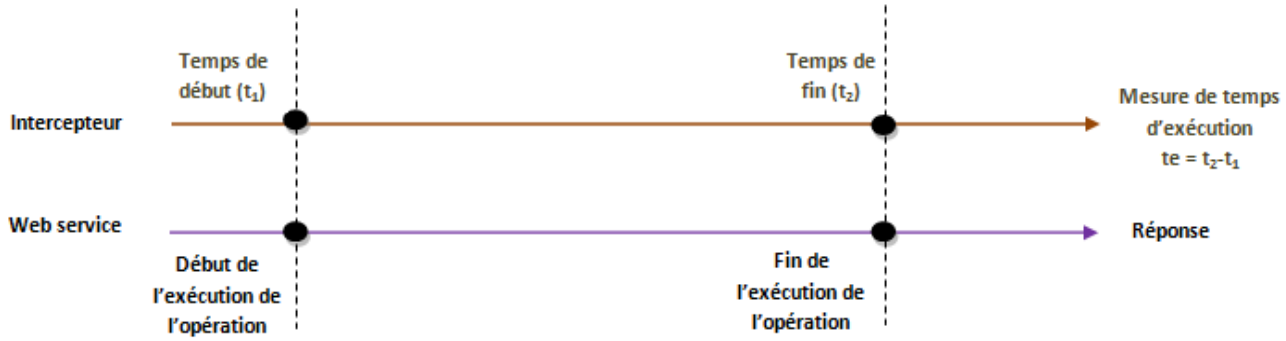

Figure 22 : Intercepteur de Monitoring de temps d'exécution.

6. Conclusion :

Dans ce chapitre, nous avons présenté un aperçu général sur notre approche, nous avons exposé les différents modèles de composition des services web, nous avons montré les formules de Monitoring des QdS d'un seul service puis d'un ensemble de services web composés, et nous conclurons notre chapitre par une représentation de la méthode de mesure utilisée, dans le prochain chapitre nous allons présentés plus de détails sur l'application développée, ainsi que les technologies que nous avons utilisé lors de l'implémentation.

Chapitre 4 : Etudes de cas et implémentation

1. Introduction :

Dans le chapitre précédent, nous avons présenté notre approche et les différentes métriques de mesure qui combinent plusieurs modèles de composition des services. Dans le présent chapitre nous nous intéressons à l'application développée afin de valider l'approche proposée, nous commençons d'abord par la représentation des différentes technologies liées à l'implémentation, ensuite nous examinons notre approche par rapport à deux études de cas, nous présentons à chaque fois le processus métier et les résultats de Monitoring appliqué.

2. Représentation des technologies utilisées :

2.1. Java :

Pour le langage de programmation, nous avons choisi java, et cela parce que java est un langage orienté objet qui possède une excellente portabilité, il peut être très facilement portable sur plusieurs systèmes d'exploitation tel que *Windows*, *Unix*, *Mac OS* et autres. Il permet de développer de nombreuses sortes de programmes [25]: des applications client-serveur, des applets qui sont des programmes java incorporés à des pages web, des servlets[10]...etc. Enfin java possède une bibliothèque de classes riche comprenant diverses fonctions parmi lesquelles on trouve des fonctions standards, un système de fichiers, et beaucoup d'autres fonctionnalités.

2.2. EJB-AOP :

La programmation orientée aspect est une technique transversale indépendante de tout langage de programmation, elle peut être mise en œuvre par n'importe quel langage qui possède un tisseur d'aspects. Pour le langage java, il existe plusieurs implémentations de l'AOP : AspectJ, SpringAOP, EJB-AOP, AspectWerkz, …etc.

Notre choix est porté sur EJB-AOP. EJB (Enterprise JavaBeans) est [27] une architecture de composants logiciels côté serveur pour la plate-forme de développement Java EE[11]. La version 3.0d'EJB prend en charge la fonctionnalité de l'AOP en fournissant la possibilité d'intercepter des méthodes métier et des rappels de cycle de vie, cette implémentation est basée sur la notion d'intercepteur (*en anglais, Interceptors*), un intercepteur est une méthode qui sera exécutée selon deux événements [26]:

[10] C'est une classe java qui permet de créer dynamiquement des données au sein d'un serveur HTTP [25].
[11] Java Plateform Enterprise Edition : spécification de la plate-forme java destinée aux applications d'entreprises.

- Intercepteur pour des événements liés au cycle de vie de l'EJB : exécuté lorsque certains événements liés au cycle de vie de l'EJB surviennent, tel que la création, la destruction, la passivation ou la réactivation.
- Intercepteur pour l'invocation des méthodes métiers : c'est une méthode qui intercepte l'appel d'une méthode métier, appelée automatiquement juste avant l'appel des méthodes métiers d'un bean session ou message driven bean.

Dans le monde de l'AOP, l'interception a lieu à différents points (points de coupures), y compris au début d'une méthode, à la fin d'une méthode, et lorsqu'une exception est déclenchée. Les intercepteurs d'EJB sont basés sur «*Around invoke advice*», ils déclenchent au début d'une méthode et restent autour lorsque la méthode revient. Ils peuvent inspecter la valeur de retour de la méthode ou toute exception lancée par la méthode. Nous avons utilisés ces intercepteurs pour obtenir des informations sur l'exécution des méthodes liées aux services web (le monitoring).

2.3. BPEL :

Pour le choix de langage de composition des services web, on a opté pour BPEL (Business Process Execution Language). C'est une représentation XML utilisée dans le cadre de la mise en place d'une architecture orientée service (SOA) en entreprise.

Nous avons choisi le langage BPEL parce qu'il facilite l'invocation des opérations des web services que se soit des opérations synchrones ou asynchrones. On peut invoquer les opérations de manière séquentielle ou parallèles, comme on peut attendre les callbacks. BPEL fournit un vocabulaire riche pour le traitement des erreurs (*fault handling*). Les fonctionnalités les plus importantes qu'offre BPEL sont [28]:

- Décrire la logique du processus métier à travers la composition de services.
- Composer des processus métiers complexes à partir de processus et de services plus simples.
- Manipuler des invocations synchrones et asynchrones des opérations des services, et gérer les callbacks qui viendront après.
- Invoquer les opérations des services en séquence ou en parallèle.
- Maintenir des activités longues qui sont interruptibles.
- Reprendre des activités interrompues ou qui ont échoué pour minimiser le travail à refaire.
- Router les messages entrants à l'activité ou au processus destinataire.
- Exécuter les activités en parallèles.
- Présenter le processus comme un service (invisible pour l'utilisateur).

3. Etudes de cas :

Pour valider et illustrer notre travail, nous présentons deux scénarios de deux différents exemples, le premier scénario représente un cas d'utilisation d'un calcul mathématique pour montrer les métriques de monitoring des services web composés indiqués précédemment, et le deuxième scénario représente une étude de cas pour la réservation de voyage : « Travel Reservation ».

3.1. Service de calcul mathématique :

3.1.1. Description de processus :

Le premier exemple que nous avons développé combine nombreux modèles de composition des services web, il représente un processus qui invoque des services pour le calcul de plusieurs opérations mathématiques, il prend en entrée deux paramètres de type réel et retourne une seule valeur réelle. Ce processus permettant le calcul de l'opération mathématique suivante :

$$\frac{\left(\left(\frac{x}{y}\right) + \left((x+y)+(x-y)+(x\times y)\right)\times\left(\frac{x}{y}\right)\right)^{3} + \left((x+y)-\left(\frac{x}{y}\right)\right)^{2}}{y}$$

La formule représente un service complexe qui expose cinq opérations mathématiques : l'addition, la soustraction, la multiplication, la division, et la puissance. Une description de ce service est représentée dans le schéma de la figure 23.

Comme on peut le voir dans le schéma, le service combine différents modèles de compositions, le premier bloc d'opérations comprend un modèle de choix exclusif qui définit une condition sur la valeur de *y* (différente de 0), le deuxième bloc inclut deux modèles de parallélisme et deux modèles de synchronisation et deux séquences, et le troisième bloc d'opérations comprend un motif de synchronisation, un motif de parallélisme et deux séquences. Les services impliqués dans le processus sont :

Add-WS : Service d'addition de deux nombres réels.

Soust-WS : Service de soustraction de deux nombres réels.

Mult-WS : Service de multiplication de deux nombres réels.

Div-WS : Service de division de deux nombres réels.

Cube-WS : Service de calcul de cube d'un nombre réel.

Square-WS : Service de calcul de carré d'un nombre réel.

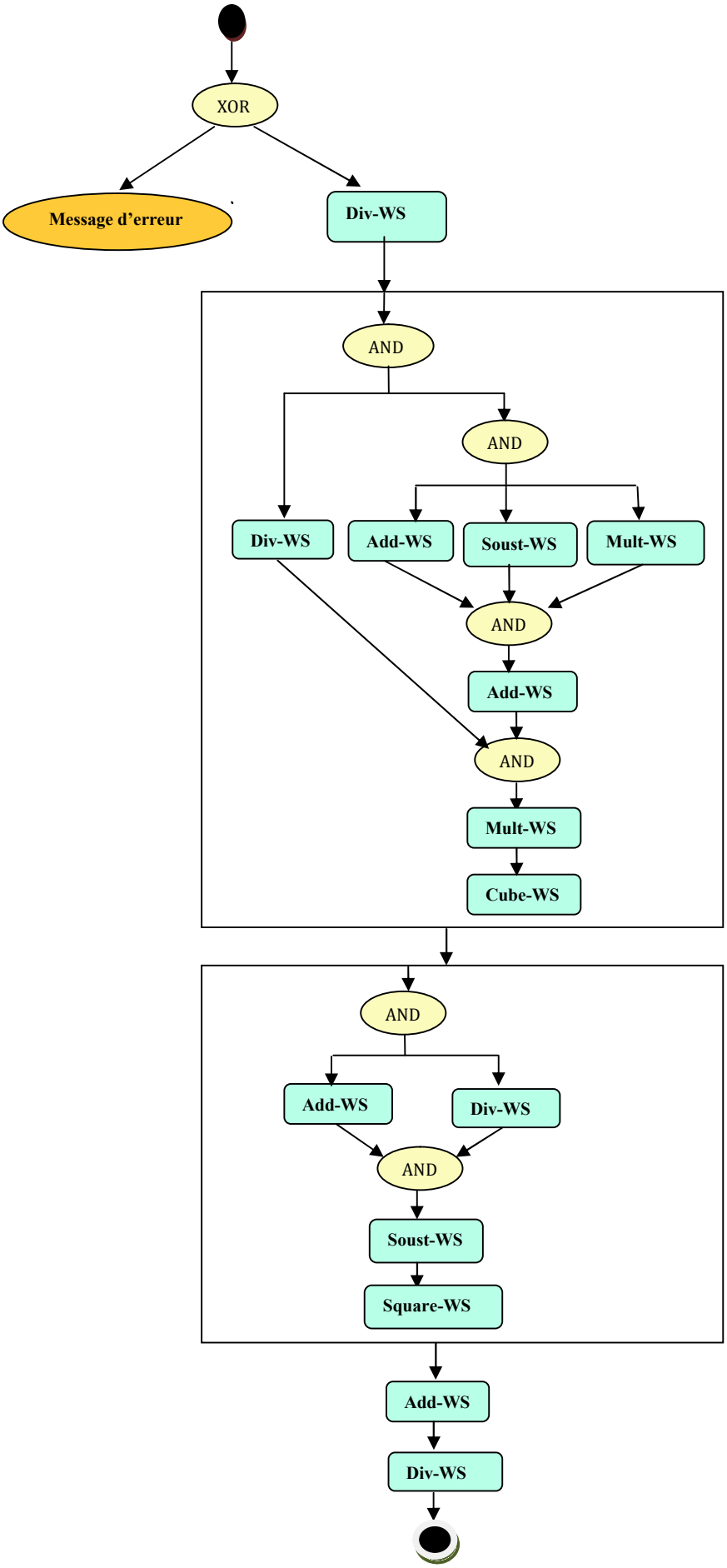

Figure 23 : Service de calcul mathématique.

3.1.2. Présentation de processus sous BPEL :

La définition d'un processus métier dans BPEL signifie essentiellement de créer un nouveau service web qui fait appel à un ensemble de services web existants. Nous avons crée et déployé les services web considérés sous le serveur *JBoss EAP*, chaque service expose une opération mathématique pour le calcul de la formule indiquée précédemment.

Après le déploiement des services web participés, nous avons importé les fichiers WSDL correspondant aux services, puis défini un processus BPEL synchrone et les différents liens partenaires pour l'interaction avec les services. L'architecture du processus est représentée dans la figure suivante :

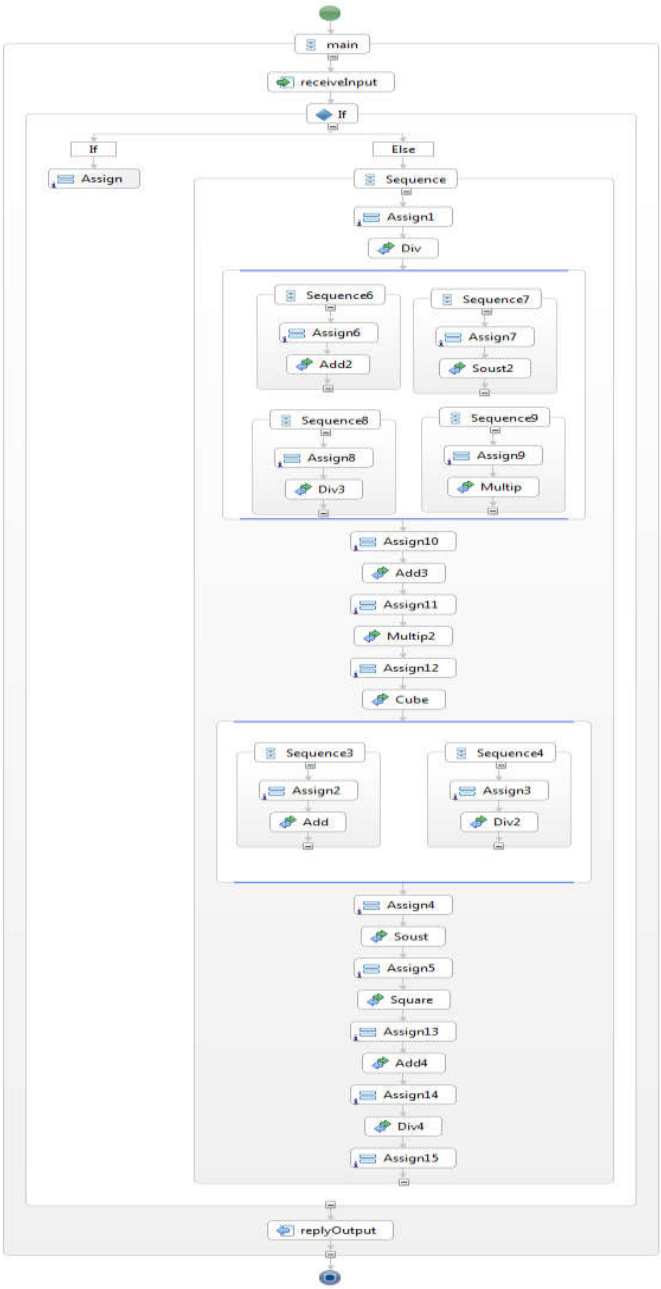

Figure 24 : Service de calcul mathématique : Présentation en BPEL

3.1.3. Application des mesures de monitoring sur les services :

Comme nous avons indiqué dans le chapitre précédent, notre principe consiste à mesurer les QdS de chaque service web séparément au niveau orienté aspect, puis à combiner les mesures pour obtenir le monitoring global de processus. Pour se faire, nous avons développé un code intercepteur qui sert à calculer le temps d'exécution, la disponibilité de chaque service puis sauvegarder ces métriques dans la base de données. Le code source de l'intercepteur est représenté dans la figure suivante :

```
public class LoggingService {

    @AroundInvoke
    Object monitorService(InvocationContext ctx) throws Exception {
        //Count the number of invocations
       //Save the time before
            try {
               return ctx.proceed();
             }
             finally {
                 // Save the time after
                // Count the successful invocations
               // Measure the execution (response) time
              // Measure the availability
             // Save the values in the database
             }
    }
}
```

Figure 25 : Code source de l'Aspect de Monitoring.

3.1.4. Composition des métriques :

Pour combiner les métriques de monitoring de QdS, nous avons développé un service web spécifique comprenant un ensemble de méthodes, ce service sera invoqué par le processus pour récupérer les valeurs des QdS de la base de données et calculer les métriques de composition de chaque bloc d'opérations selon le modèle de composition des services, les valeurs de monitoring globales seront retourné à la fin par le processus.

3.1.5. Résultats de monitoring :

Les attributs de temps d'exécution et disponibilité sont mesuré du coté serveur, pour le temps de réponse, nous avons effectué les mesures du coté client, nous avons développé et déployé une application web cliente sous le serveur *JBoss EAP* avec une interface JSP. L'interface cliente est représentée dans la figure 26 :

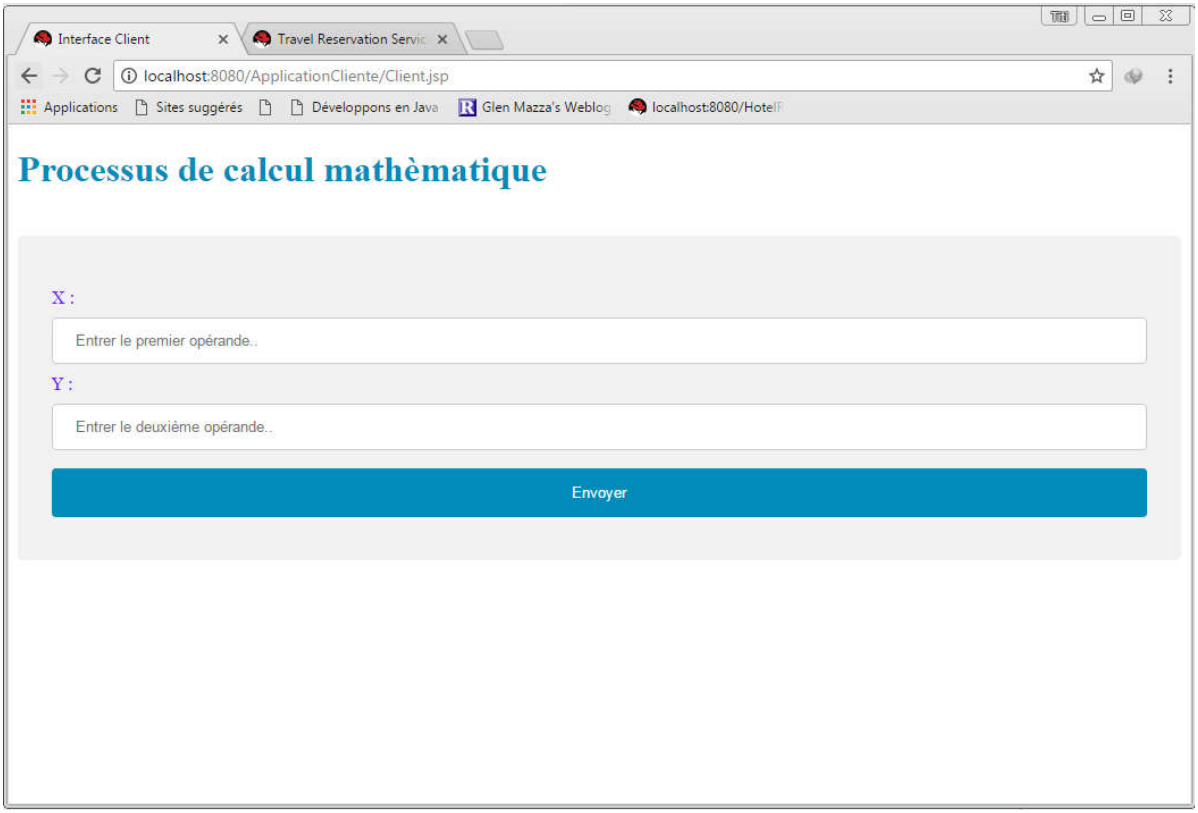

Figure 26: Processus de calcul mathématique : Interface Cliente.

L'application consiste à invoquer le processus métier et recevoir la réponse ainsi que les résultats de monitoring, et mesurer le temps de réponse global du processus. Le résultat de Monitoring est représenté dans la figure 27.

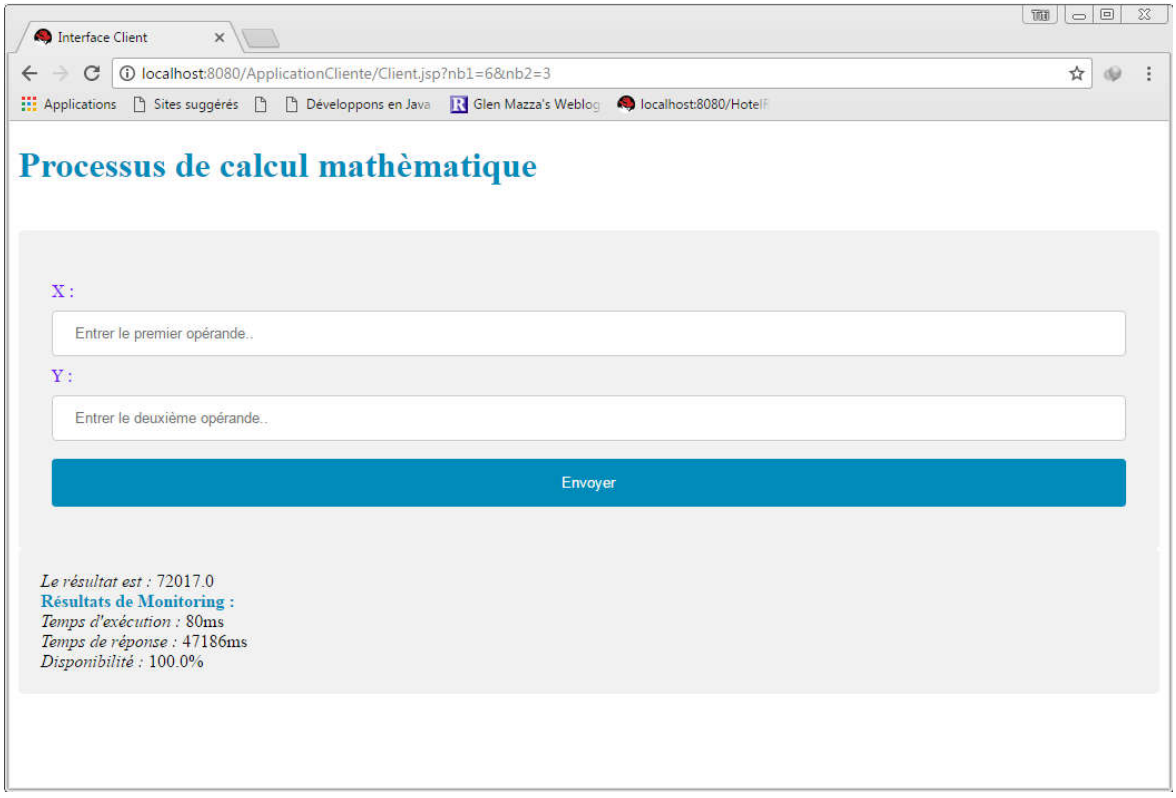

Figure 27 : Processus de calcul mathématique : Résultats de Monitoring.

3..2. Service de réservation de voyage :

3.2.1. Description de processus :

Le deuxième exemple représente un service de réservation de voyage (en anglais, Travel Reservation Service). Nous montrons qu'un client fait une demande pour acheter un billet d'avion, réserver dans un hôtel et louer une voiture, le processus de voyage doit communiquer avec un service aérien, un service hôtelier et un service de location de voitures pour répondre à la demande du client, le processus est montré dans la figure 28:

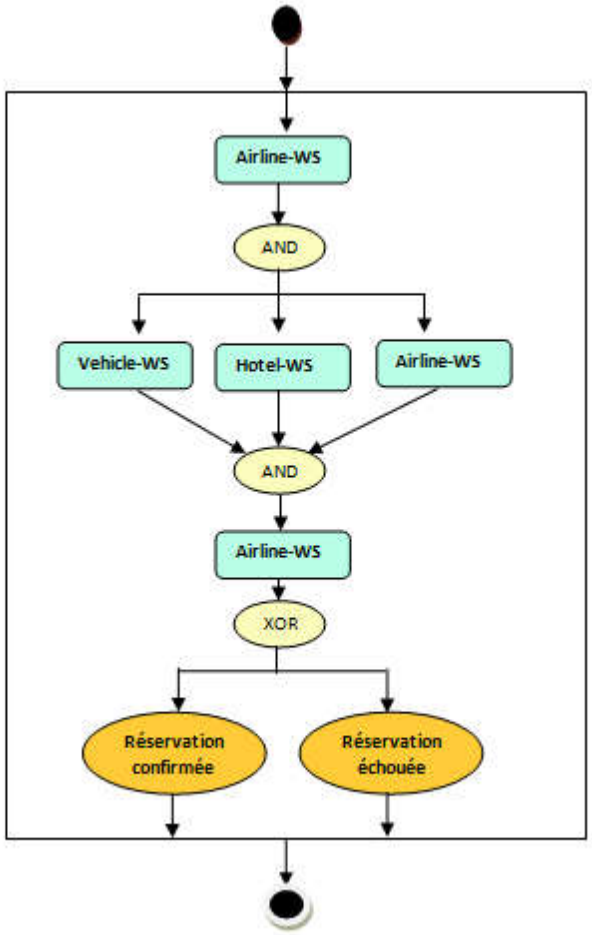

Figure 28 : Processus de réservation de voyage

Airline WS : service de réservation de vol.

Hotel WS : service de réservation de la chambre d'hôtel.

Vehicle WS : service de location des voitures.

3.2.2. Présentation de processus sous BPEL :

Le processus du deuxième exemple est crée aussi dans le langage BPEL, pour être déployé par la suite par le serveur Apache ODE. Les services partenaires sont crées et déployés sous le serveur JBoss EAP, nous avons appliqués les mesures de monitoring sur

chaque service seul de la même manière que l'étude de cas précédente, le processus BPEL est représenté dans la figure 29.

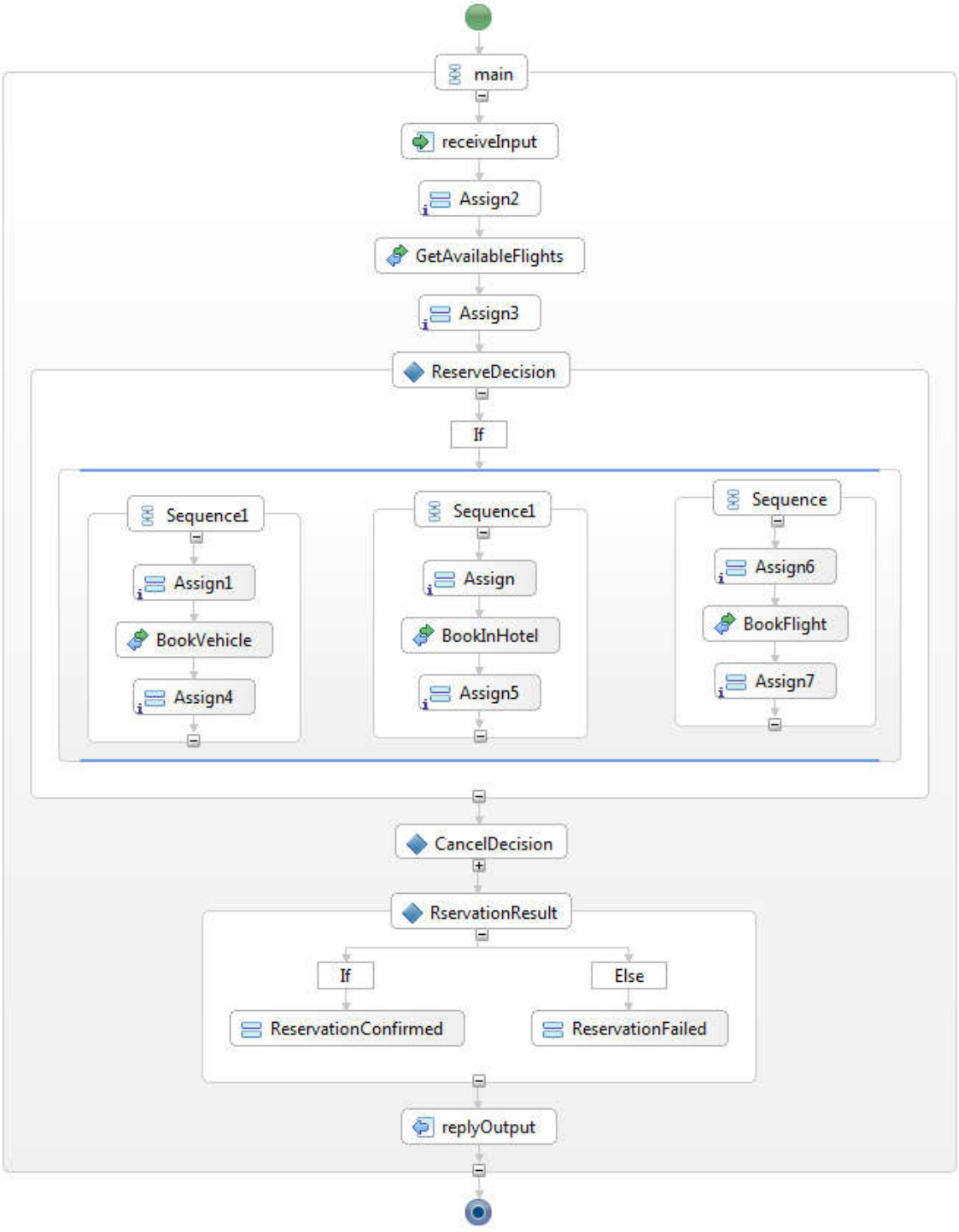

Figure 29 : Processus de réservation de voyage : présentation en BPEL.

3.2.3 Résultats de monitoring :

Nous avons appliqué l'approche proposée sur cette étude de cas de la même manière que nous l'avons appliqué sur l'étude de cas précédente. Pour mesurer le temps de réponse et voir les résultats de monitoring, nous avons crée et déployé une application web cliente sous le serveur *JBoss EAP* avec une interface JSP, elle se présente dans la figure 30.

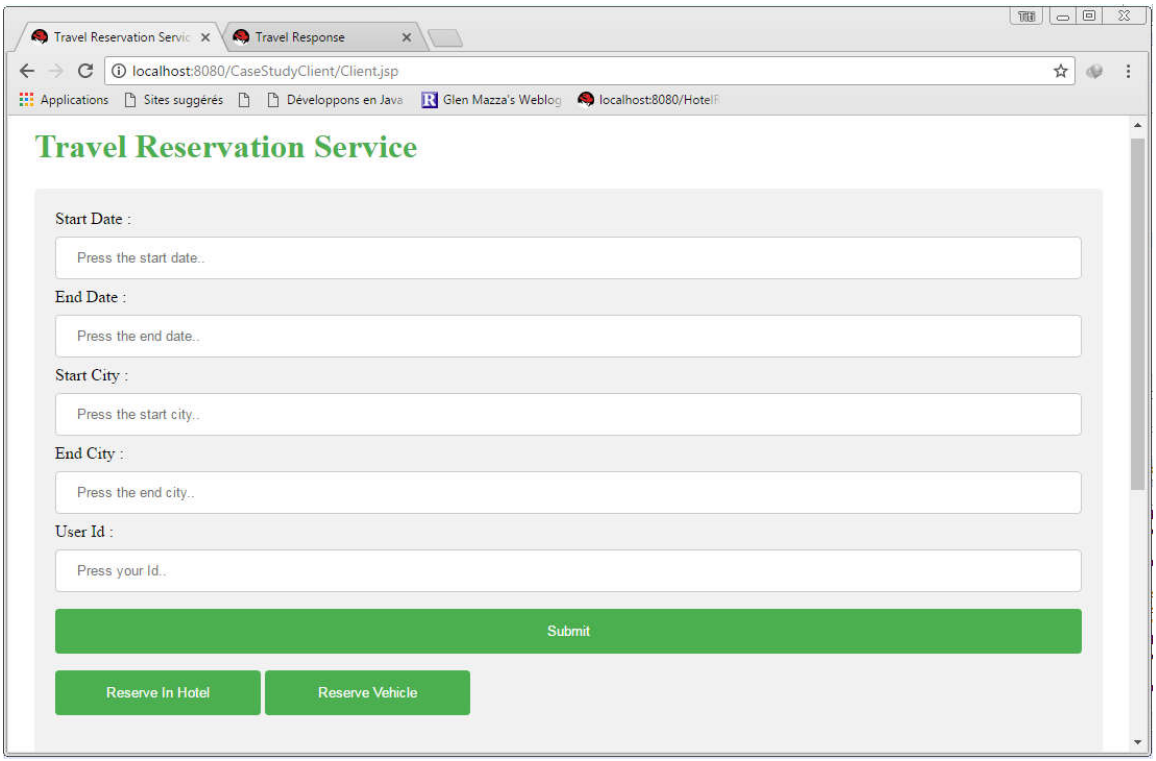

Figure 30 : Processus de réservation de voyage : Interface Cliente

Comme on peut le voir dans la figure 30, le client doit remplir le formulaire de réservation pour avoir la liste des compagnies aériennes disponibles, après l'envoi de la requête, le résultat de monitoring est affiché au client comme le montre la figure suivante :

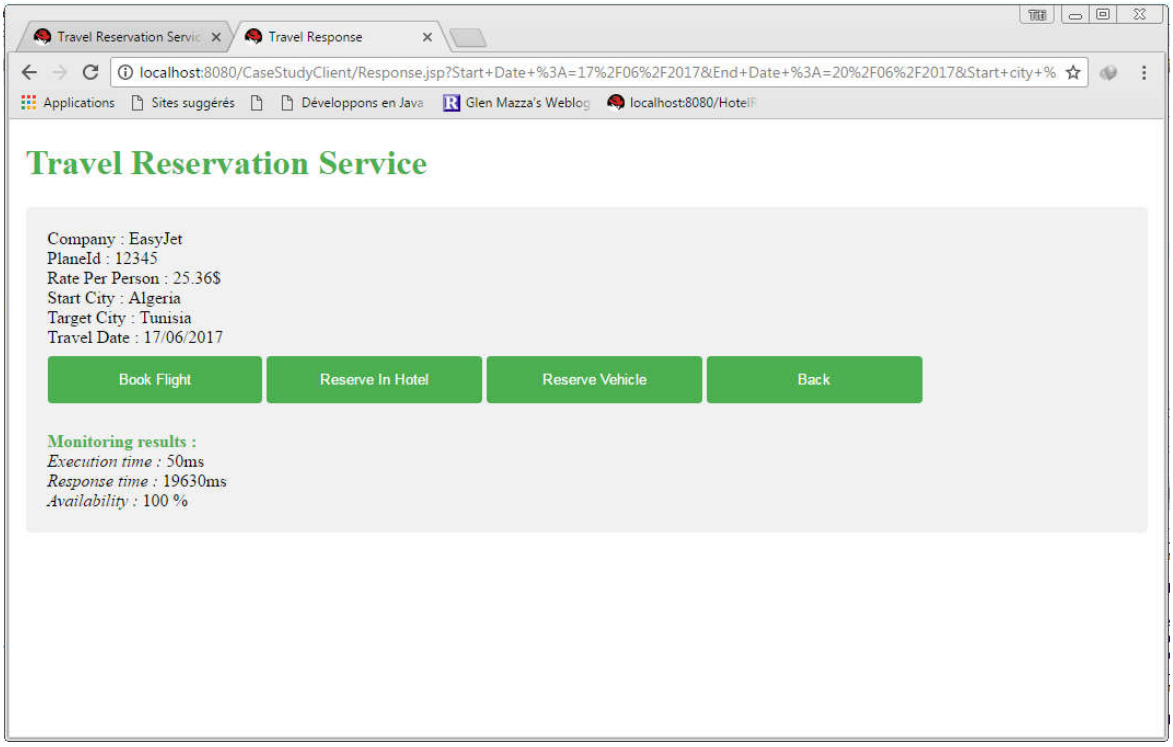

Figure 31 : Processus de réservation de voyage : Résultats de Monitoring.

4. Conclusion :

Ce chapitre est dédié à la présentation de l'application développée pour résoudre le problème de monitoring des QdS des services web composés. Au premier temps, nous avons présenté les différentes technologies utilisées lors de l'implémentation, puis nous avons examiné notre approche par rapport à deux études de cas en présentant pour chacune le processus métier développé. A la fin nous avons illustré les résultats de Monitoring sur les deux exemples par des captures d'écran.

Conclusion générale et perspectives

Conclusion générale et perspectives:

Les services web représentent une technologie essentielle pour l'implémentation des applications distribuées, ils permettent aux entreprises et individus de publier des liens vers leurs données et leurs applications de la même manière qu'ils publient des liens vers leurs pages web. La composition des services web, notamment, est considérée comme un point fort car elle apporte une solution au problème des besoins complexes du client.

Dans ce mémoire, nous avons proposé une approche pour le Monitoring de QdS des services web composés au sein d'un processus métier ; l'idée principale de cette approche consiste à mesurer les QdS de chacun des services web impliqués dans la composition en utilisant les intercepteurs de la programmation orientée aspect, puis combiner les métriques obtenues selon différents modèles de composition afin de réaliser un monitoring global du service composé. Dans notre travail, nous avons pris en considération trois types de QdS à savoir le temps d'exécution, le temps de réponse et la disponibilité.

Afin de montrer la pertinence de notre approche, nous l'avons appliqué sur deux études de cas, la première porte sur la réalisation d'un calcul mathématique complexe réalisé par un ensemble de services web tandis que la deuxième porte sur un service web composé pour l'organisation d'un voyage/séjours.

La réalisation de notre approche nous a permis de dégager quelques perspectives de travail à savoir la prise en charge d'autres types de QdS notamment les QdS complexes et les QdS qualitatives. Une QdS est dite qualitative si elle ne peut être mesurée comme les QdS liées à la sécurité.

Bibliographie

Bibliographie :

[1] : Hajar Omrana. Vers une Composition Dynamique des Services Web : une approche de Composabilité Offline. Thèse de Doctorat. Rabat : Université Mohammed V–AGDAL, 2014, 194p.

[2] : Riadh BEN HALIMA. Conception, implantation et expérimentation d'une architecture en bus pour l'auto-réparation des applications distribuées à base de services web. Thèse de Doctorat. Toulouse : Université de Toulouse, Jeudi 14 Mai 2009, 118p.

[3] : Y.BELAID. Services web (SOAP). Fiche de lecture, Urbanisation des SI-NFE107.

[4] : Adolphe Francois, Julien Marmel, Dominique Perlat, Olivier Printemps. SOAP : Simple Object Access Protocol. 37p.

[5] : Yousri Kouki, Damián Serrano, Thomas Ledoux, Pierre Sens, Sara Bouchenak. SLA et qualité de service pour le Cloud Computing. Paris : Université de Grenoble I – LIG, Grenoble, du 16 au 18 janvier 2013,11.

[6] : Enrique Lafuente Hernandez. Evaluation Framework for Quality Of Service in web Services : implementation in a pervasive environment. Lyon : Institut National de Sciences Appliquées (INSA), 40p.

[7] : Chrysostomos Zeginis. Monitoring the QoS of Web Services using SLAs – Computing metrics for composed services. Crete : University of Crete, March 2009,102.

[8] : Mani Anbazhagan and Nagarajan Arun. Understanding quality of service for web services. https://www.ibm.com/developerworks/library/ws-quality/. 01 January 2002.

[9] : Chemidi Zoulikha. Sélections des services web à base de colonies de fourmis. Tlemcen: Université Abou Bekr Belkaid, le 02 juillet 2012,66p.

[10] : Liangzhao Zeng, Hui Lei, Henry Chang. Monitoring the QoS for Web Service.http://link.springer.com/chapter/10.1007/978-3-540-74974-5_11#page-1.2007.

[11] : Jael Zela Ruiz, Cecilia M. Rubira. Quality of Service Conflict During Web Service Monitoring: A Case Study. Sao Paulo, Brazil : University of Campinas, 2016.

[12] : Zekri Oumia. Conception et réalisation d'un modeleur de composition de service web SOAP et REST. 04 Janvier 2015.

[13] : Nicolas Repp, Rainer Berbner, Oliver Heckmann, and Ralf Steinmetz. A cross-layer Approach to Performance Monitoring of Web Services. multimedia Communications Lab (KOM). TechnischeUniversitat Darmstadt-Germany.

[14] : Florian Rosenberg, Christian Platzer, and SchahramDustdar. Bootstrapping Performance and Dependability Attributes of Web Services. Vienna University of Technology.

[15] : Jean Baltus, La Programmation Orientée Aspect et AspectJ : Présentation et Application dans un Système Distribué, Facultés Universitaires Notre-dame de la Paix, Namur, Belgique ,2002.

[16] : Philip Bianco, Grace A. Lewis et Paulo Merson, Service Level Agreements in Service-Oriented Architecture Environments, Technical note CMU/SEI-2008-TN-021, Software Ingineering Institute. Septembre 2008.

[17] : Heiko Ludwig, Alexander Keller,Asit Dan, Richard P. King, Richard Franck. Web Service Level Agreement (WSLA) Language Specification, IBM Corporation, 2001, 2002, 2003.

[18] : Li - jie Jin, Vijay Machiraju, AkhilSahai. Analysis on Service Level Agreement of Web Services, Software Technology Laboratory, HP Laboratories Palo Alto, 21 June 2002.

[19] : Amina BEKKOUCHE, Composition des Services Web Sémantiques À base d'Algorithmes Génétiques, Tlemcen : Université Abou-bekr Belkaid, 2012, 119p.

[20] : Chris Peltz, Web Services Orchestration and choreography, 2003, p46.

[21] : Hernan Dario ROJAS. Orchestration a haut niveau et BPEL. Thèse de Master. France : Ecole Doctorale Mathématiques, Sciences et Technologies de l'information, Informatique, 2006, 88p.

[22] : Haiteng Zhang, Zhiqing Shao, Hong Zheng, and Jie ZhaiWeb, Service Reputation Evaluation Based on QoS Measurement, Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China, publié le 13 avril 2014. Disponible sur :https://www.hindawi.com/journals/tswj/2014/373902/

.

[23] : KEBIR, Selim. « Programmation orientée aspect en Java avec AspectJ », 2 février 2010, http://skebir.developpez.com/tutoriels/java/aspectj/.

[24] : Brakni Ilhem. Planification multi-agent pour la composition dynamique. Thèse de Master. Université de Tébessa, 2009-2010, disponible sur : http://www.memoireonline.com/08/11/4782/m_Planification-multi-agents-pour-la-composition-dynamique0.html .

[25] : Jean-Michel DOUDOUX. Présentation de Java, Disponible sur : https://www.jmdoudoux.fr/java/dej/chap-presentation.htm .

[26] : BEN BRAIEK HOUSSEM, JEBNOUN HADHEMI. La programmation Orientée Aspects AOP.

[27] : Jean-Michel DOUDOUX. Les EJB (Entreprise Java Bean). Disponible sur : https://www.jmdoudoux.fr/java/dej/chap-ejb.htm .

[28] : Stephen Gilmore. Enterprise Computing : Business Process Execution Language (BPEL), Université de Edinburgh, 83p, Disponible sur : https://fr.scribd.com/document/306792092/Bpel-Study-Material .

[29] : Red Hat JBoss Developer Studio, Overview, Disponible sur : https://developers.redhat.com/products/devstudio/overview/?referrer=jbd .

[30] : Issam RABHI. Testabilité des services web. Thèse de Doctorat. France : Université Blaise Pascal – Clermont II, 2012, 115p.

[31] : Apache Software Foundation, Apache ODE : About Apache ODE, Disponible sur : http://ode.apache.org/index.html .

[32] : H2 Database Engine, Disponible sur : http://www.h2database.com/html/main.html .

[33] : Niko Thio and Shanika Karunasekera. Automatic measurement of a qos metric for web service recommendation. In ASWEC ’05 : Proceedings of the 2005 Australian conference on Software Engineering, Washington, DC, USA, 2005. IEEE Computer Society.

Annexes

Annexes :

Les outils utilisés :

1. Red Hat JBoss Developer Studio :

JBoss Developer Studio (JBDS) est un environnement de développement créé et développé actuellement par JBoss. Il intègre et certifie à la fois des outils et des composants d'exécution en combinant Eclipse, Eclipse Tooling et JBoss EAP (Enterprise Application Platform). Les outils de développement intégrés sont utilisés pour créer des applications Web riches utilisant des technologies open source telles que JBoss Seam[12], JBoss Application Server, Hibernate[13] et JBoss jBPM [29].

Red Hat JBoss Developer Studio fournit une assistance supérieure sur l'ensemble de cycle de vie d'un développement en un seul outil, Il s'agit d'un ensemble de développement intégré (IDE) basé sur Eclipse, certifié pour développer, tester, et déployer des applications web riches, des applications web mobiles, des application d'entreprise transactionnelles et des applications d'intégration basée sur une architecture orientée service (SOA).

JBoss Developer Studio est constamment mis à jour pour inclure les dernières versions d'Eclipse et Web Tools Project (WTP) et fournit des outils pour Java EE et le développement web, comme[29]:

- Java EE, JSF et JSP tools.
- JPA tools.
- Server tools.
- Web services et WSDL tools.
- HTML, CSS, et JavaScript tools.
- XML, XML Schema et DTD tools.

[12] Un framework d'application web développé par JBoss, http://seamframework.org/
[13] Est un framework open source gérant la persistance des objets en base de données relationnelle. http://hibernate.org/

2. JBoss EAP (JBoss Enterprise Application Plateform) :

JBoss Enterprise Application Plateform[14] est une plate-forme d'exécution du serveur d'applications, open source, basée sur Java EE, et qui s'utilise pour concevoir, gérer, déployer et héberger des applications et services java hautement transactionnels. C'est une plate-forme d'application basée sur une architecture souple et modulaire aux composants axés sur les services. Elle simplifie le développement des applications dans différents environnements [29].

Parce qu'il est basé sur Java, le serveur d'applications JBoss fonctionne sur plusieurs plates-formes; Il est utilisable sur n'importe quel système d'exploitation qui support Java.

3. Apache ODE :

Apache ODE[15] (Apache Orchestration Director Engine) est [31] un logiciel qui exécute un ou plusieurs processus métiers qui on été exprimés dans un langage BPEL. Il communique principalement avec un ou plusieurs services Web, envoie et réception des messages, manipule des données et gère des exceptions («erreurs»). Le moteur est capable d'exécuter des processus de longue et de courte durée pour coordonner tous les services web qui composent un service ou une application (orchestration).

Le moteur ODE a deux couches de communication, avec lesquelles il interagit avec le monde extérieur [31] :

- ✓ Couche d'intégration Apache Axis2[16]: prend en charge la communication avec les services Web.
- ✓ Couche basée sur la norme JBI[17]: prend en charge la communication via les messages JBI.

[14] https://developers.redhat.com/
[15] http://ode.apache.org/
[16] http://axis.apache.org/, un logiciel créé pour faciliter le développement des services web en utilisant la technologie SOAP.

4. H2 Database Engine :

H2[18] est un système de gestion de base de données Java SQL, open source, il peut être intégré à une application Java ou bien fonctionner en mode client-serveur, les principaux caractéristiques de H2 sont [32]:

- Très rapide, open source.
- Il propose des interfaces de programmation (APIs) SQL et JDBC.
- Base de données en mémoire : les tables peuvent être crées en mémoires vives ou sur disque, elles peuvent être persistantes ou temporaires.
- Application de la console basée sur le navigateur.
- Fichier Jar de petite taille : environ 1 Mo.

[17] Java Business Integration : une spécification développée sous JCP (Java Community Process) pour une approche de la mise en œuvre d'une application orientée service.

[18] http://www.h2database.com/html/main.html