MASTER EN SCIENCES INFORMATIQUES

Substitution de web services

Etat de l'art

Masure, Gilles

Award date:
2013

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur
Faculté d'Informatique
Année académique 2012–2013

La substitution de web services
Etat de l'art

Gilles Masure

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Philippe Thiran

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Facultés Universitaires Notre-Dame de la Paix de Namur
Institut d'Informatique
Rue Grandgagnage, 21
B-5000 Namur

Année académique 2012-2013

La substitution de web services

Etat de l'art

Gilles Masure

Mémoire présenté en vue de l'obtention du grade de master en informatique.

Remerciements

J’ai le plaisir de consacrer cette section à remercier tous ceux qui ont contribué à la réalisation de ce travail.

Je tiens, tout d’abord, à remercier mon promoteur, Monsieur Thiran, professeur à la faculté des sciences informatiques de l’université de Namur, pour ses conseils, ses remarques et pour m’avoir remis sur les bons rails quand cela s’est avéré nécessaire.

Merci aussi à Madame Delheixe et à mes parents pour leur relecture attentive.

Enfin, j’exprime un remerciement particulier à Sophie De Waegeneer pour son soutient indéfectible pendant ces années d’études.

Table des matières

Table des figures

1. Introduction

Il commence à être révolu le temps où chaque société développait en interne les logiciels dont elle avait besoin.

Pour des raisons de compétence et d'économie, ce sont de plus en plus souvent des sociétés spécialisées qui fournissent les logiciels.

Ces mêmes raisons d'économie ont fait prendre conscience qu'un grand nombre de traitements sont répétitifs et peuvent être réalisés par le même module. Si on extrapole cette constatation à un ensemble de sociétés, on obtient l'idée de base des web services : des « applications métier modulaire qui sont ouvertes et basées sur des standards » (UDDI 01).

En raison de leurs nombreux avantages : langage standard de communication, facilité de maintenance, interopérabilité, …, les web services se développent depuis plusieurs années de façon rapide et sont désormais un modèle business à part entière : des entreprises gagnent de l'argent en faisant payer l'accès à leur web service.

Ce modèle business des web services implique que leurs utilisateurs aient développé des exigences de qualité sur ces services : je ne suis prêt à payer un certain prix que si j'ai la garantie d'avoir un service de qualité.

Dans ce cadre, il est intéressant de se pencher sur la problématique de la substitution des web services. Soit « remplacer un composant par un autre composant pour autant que le composant remplaçant produise le même output et satisfasse aux mêmes besoins que le composant remplacé.». (Taher & al. 09).

La substitution permet d'apporter une réponse à ces questions :

Dois-je considérer comme une fatalité le fait de ne plus pouvoir accéder à la fonctionnalité offerte par le web service que j'utilise, du fait de son indisponibilité ? Dois-je accepter que le fournisseur du service baisse la qualité offerte ? M'est-il possible de remplacer un web service par un autre offrant une meilleure qualité de service ?
Pour pouvoir répondre à ces questions et choisir en connaissance de cause quel web service remplacera celui à substituer, il nous faut prendre en compte deux critères : les aspects fonctionnels et non-fonctionnels.

Concernant les aspects fonctionnels, il est essentiel de trouver un autre web service qui rende le même service. Pour faire un choix sur ces critères fonctionnels, il nous faut d'abord comprendre ce qu'est exactement un web service, quel est son mode de fonctionnement et où l'on peut trouver ces

informations sur son fonctionnement.

Concernant les aspects non-fonctionnels, l'enjeu dans le cadre de la substitution est de pouvoir se baser sur un autre critère de comparaison si des web services ont des aspects fonctionnels équivalents. Les aspects non-fonctionnels qui sont pris en compte sont les qualités de service. Un aspect qualité a une influence plus directe que les autres sur la substitution : il s'agit de la disponibilité. Si un web service n'est plus disponible, il faut lui trouver un substitut. Nous verrons qu'il existe d'autres aspects de qualité qui sont pris en compte si on désire substituer le web service que l'on utilise par un autre offrant une meilleure qualité de service. Notons qu'il existe d'autres aspects non-fonctionnels que la qualité de service, tel que le fournisseur du service, mais nous ne les aborderons pas dans le cadre de ce mémoire car ce n'est pas pertinent pour la substitution.

Une fois ces aspects mis en place, nous pourrons entrer dans le vif du sujet et aborder la problématique de la substitution.

Cette substitution peut être obligatoire, c'est le cas si le service appelé n'est pas disponible. La substitution peut également être voulue afin d'obtenir une meilleure qualité de service. Pour trouver des solutions au problème de la substitution pour cause d'indisponibilité, nous commencerons par voir la substitution interne ou comment un fournisseur de service peut assurer la disponibilité de son propre service.

Nous aborderons ensuite la problématique de la substitution externe, ou comment faire lorsqu'un web service n'est plus disponible ou ne répond plus à nos exigences de qualité pour trouver un autre service capable de répondre à notre requête et quels en sont les enjeux. Les premières méthodes que nous vous décrirons permettent, comme c'est le cas dans la substitution interne, de faire face à une perte de disponibilité. Nous présenterons ensuite des méthodes qui permettent non seulement de faire face à une perte de disponibilité mais aussi de prendre en compte les autres aspects de la qualité de service.

Nous serons alors en mesure de vous proposer une évaluation synthétique qui permettra de comparer toutes ces méthodes. La discussion que nous aurons pour terminer nous permettra de répondre à cette question : quelle substitution utiliser, pour rencontrer quel besoin?

2. Les web services

Dans ce chapitre nous allons commencer par définir les web services, voir quelle est leur utilité et comprendre quels en sont les éléments constitutifs.
Nous pourrons alors aborder des notions plus avancées comme la qualité dans les web services.
Voir toutes ces notions est indispensable pour poser le cadre dans lequel s'exécute la substitution.

2.1 Définition et utilité des web services

Prenons l'exemple d'un web master d'un site d'information sur le championnat de Belgique de football qui souhaite informer les visiteurs des résultats des matchs du week-end, donner des statistiques sur les buts, les passes, donner la possibilité de réserver des places, proposer certains résumés et d'autres services à valeur ajoutée. Les défis auxquels il va devoir faire face sont immenses : faire appel à des acteurs différents qui vont lui permettre de rendre ces services. Tous ces acteurs utilisent des applications différentes, écrites dans des langages de programmation particuliers, mises en place sur des machines hétérogènes… Bref, du travail en perspective pour faire collaborer tous ces acteurs!

C'est là qu'intervient l'utilité des web services :
« Les services Web fournissent des moyens standard de l'interopérabilité entre différentes applications logicielles, fonctionnant sur une variété de plates-formes et / ou de frameworks. » (Booth & al. 04). Les web services vont donc permettre au programme de statistique sur les buts développé en python qui s'exécute sur une machine Windows, au programme délivrant les résultats du week-end développé en C++ qui s'exécute sur une machine Unix et au site web développé via AJAX de communiquer entre eux et de s'échanger des informations.

Ils ont en fait deux utilités principales.

D'une part, la réutilisation de composants souvent utilisés. Plutôt que toute (partie d') application ayant besoin de ce composant l'implémente, on le fait une fois pour toutes et tous les composants qui ont besoin du web service peuvent l'utiliser. Pour calculer le taux de réussite de l'attaquant vedette du FC Namur, nul besoin que tous ceux qui veulent la réponse implémentent la fonction qui le calculera : un le fait, rend son application disponible via un web service et les autres appellent ce web service.

D'autre part, ils peuvent servir à connecter des applications existantes. Les web services peuvent résoudre les problèmes d'interopérabilité en permettant à des applications de plates-formes

différentes de mettre en commun leurs données.

Voici une autre définition que l'on peut trouver dans la littérature :

« Un web service est considéré comme auto-suffisant, auto-descriptif, modulaire qui peut être publié, localisé et invoqué à travers le web» (Thirumanan & al. 10). Auto-suffisant veut dire qu'ils n'ont besoin que d'eux-mêmes pour rendre la fonctionnalité qu'ils exposent. Ils sont auto-descriptifs grâce à un document qui décrit leur fonctionnalité et la façon de les utiliser.

Les web services permettront donc à notre web master du site d'information sur le football de faire communiquer son application avec les différentes applications tierces et d'échanger les informations utiles à apporter de la valeur ajoutée au site internet.

Notons qu'un web service peut offrir plusieurs méthodes. Un même web service peut par exemple offrir, via une méthode, le nombre de buts inscrits pour une équipe donnée et, via une autre, le nombre de buts inscrits pour un joueur donné. Dans le cadre de ce mémoire, nous utiliserons, par souci de simplification, le mot « web service » pour « une opération particulière de ce web service ». Autrement dit, quand nous utilisons le mot web service, il s'agit d'un web service n'offrant qu'une seule opération. Notre web service précité n'offrant plus que la méthode pour connaitre le nombre de buts pour une équipe donnée.

Ceci posé, détaillons maintenant le fonctionnement des web services : quels sont les protocoles et langages de programmation utilisés, comment fait-on pour utiliser des web services, qu'est-ce qu'un appel statique et qu'est-ce qu'un appel dynamique et quel est le cheminement type à suivre pour utiliser un web service. Cela nous permettra de comprendre comment se fait le choix d'utiliser un web service plutôt qu'un autre en se basant sur ses aspects fonctionnels. Et cela nous permettra de comprendre par la suite la problématique de la substitution sur base de ces aspects.

2.2 Fonctionnement des web services

Les web services étant définis et leur utilité exposée, penchons-nous à présent sur leur mode de fonctionnement. Cela nous aidera à comprendre comment les utiliser. Et également quelles sont les difficultés que l'on peut rencontrer lorsque l'on désire substituer un web service par un autre rendant le même service.

Rappelons tout d'abord que les services web sont des composants d'applications qui peuvent être utilisés par d'autres applications. Ils communiquent en utilisant des protocoles ouverts.

Il faut donc que les web services se basent sur un langage compréhensible par différents langages d'applications et utilisable sur différentes plates-formes.

Ce langage c'est XML : l'e**X**tensible **M**arkup **L**angage, langage de balise conçu pour le stockage et le transport de donnée. Il est la base des trois éléments nécessaires à l'utilisation des web services (W3C 12) :

- *WSDL* (**W**eb **S**ervice **D**escription **L**anguage)
- *SOAP* (**S**imple **O**bject **A**cces **P**rotocol)
- *UDDI* (**U**niversal **D**escription, **D**iscovery and **I**ntegration)

WSDL précise quels messages doivent être échangés pour interagir avec un web service, SOAP offre les bases de la communication et UDDI est un annuaire permettant de trouver le web service dont on a besoin. Nous verrons au chapitre 2.5 un schéma permettant de comprendre comment ces notions interagissent entre elles.

Voyons dès à présent une description en couches des notions intervenant dans les web services. Cette description en couches permet de visualiser à quel moment les notions intervenant dans l'utilisation des web services agissent et avec quelles autres notions elles interagissent.

2.3 Description en couches des web services

Afin d'avoir une vue d'ensemble des interactions entre les différentes notions que nous venons de voir, nous vous présentons une description en couches des web services proposée notamment par (Booth & al. 04).

Processus Découverte, substitution, ...
Registre de service UDDI
Description WSDL
Messages SOAP
Communication HTTP, FTP, ...

Fig. 1 : Description en couches des web services.

La couche la plus basse est la couche de communication. Les web services qui sont publiés sont accessibles via un protocole de communication, le plus souvent HTTP[1] mais cela peut aussi être FTP[2] ou un autre protocole.

La couche message est mise en pratique avec l'utilisation de SOAP qui va notamment emballer les appels de méthodes vers le web service et va permettre de récupérer les résultats transportés par le protocole de communication.

La couche description fournit l'interface du service via WSDL qui indique quels sont les messages SOAP à échanger pour obtenir la fonctionnalité promise par le web service.

[1] HTTP – **H**yper**T**ext **T**ransfert **P**rotocol – est un protocole de communication client-serveur développé pour le World Wide Web (Fielding & al. 99)

[2] FTP – **F**ile **T**ransfert **P**rotocol – est un protocole de transport de transport de fichier qui observe le modèle client/serveur (D.Parker 05)

UDDI fournit des registres pour la publication et la recherche de web services (couche registre) en rendant disponible les documents WSDL descriptifs du web service.

La couche processus permet quant à elle de réaliser des opérations de plus haut niveau sur les web services. C'est la couche utilisatrice des web services, celle qui va tirer parti de leurs fonctionnalités en commençant par consulter des registres (typiquement UDDI mais nous en verrons d'autres, notamment au chapitre 3.3.2.3) pour trouver les web services correspondant à ses besoins.

Pour disposer des outils indispensables pour comprendre les mécanismes mis en œuvre dans les méthodes de gestion de la substitution, une description plus approfondie des éléments intervenant dans l'utilisation des web services est nécessaire. C'est l'objectif du chapitre suivant. Nous verrons alors un scénario d'utilisation d'un web service qui nous permettra de mieux visualiser les interactions entre les notions vues.
Nous aurons alors fait le tour des aspects fonctionnels des web services et pourrons alors aborder les aspects non-fonctionnels des services web qui nous intéressent : leurs qualités de service.

2.4 Eléments d'une architecture de web services

Comme indiqué dans le chapitre précédent, une architecture de web services est classiquement composée de trois éléments : des registres UDDI, des documents WSDL et des messages SOAP.

Décrivons-les plus en détail dans les sous-chapitres suivants.

2.4.1 WSDL

WSDL est un langage basé sur XML qui permet la localisation et la description de web services. Il permet de décrire les méthodes que le web service expose.

Selon les spécifications de la W3C[1], un document WSDL contient principalement ces éléments :

[1] W3C – World Wide Web Consortium – communauté internationale qui s'est donnée pour mission de développer les standards web (W3C 12).

<types> les types de données utilisées par le web service.

<messages> les messages utilisés par le web service.

<portType> les opérations exécutées par le web service.

<binding> Les protocoles de communication utilisés par le web service.

Fig. 2 : Eléments d'un document WSDL

Une intervention humaine est donc souvent nécessaire ici pour vérifier si un web service convient aux besoins. Le développeur voulant intégrer l'utilisation d'un web service à son programme doit lire le document WSDL le décrivant. Il doit prendre en compte les paramètres d'entrée et de sortie, lire le descriptif du web service et des méthodes qu'il propose.

Il n'est pas rare qu'un fournisseur de web services propose un « Try it » de son web service. Cela permet au développeur de s'assurer que le web service correspond au mieux à ses besoins.

Le document WSDL est donc un élément essentiel pour l'utilisation du web service.

Voici un exemple de document WSDL (sans la partie binding, qui est la partie qui permet de préciser le format des données et le protocole utilisé (W3C 12), par souci de simplification) qui présente la description d'un des services utilisés par notre web master du site de football, le nombre de buts inscrits sur une saison par un joueur :

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="jupilerLeageInfos"
        targetNamespace="http:// jupilerLeageInfos.jaxrpc.samples/"
         xmlns:tns="http:// jupilerLeageInfos.jaxrpc.samples/"
         xmlns="http://schemas.xmlsoap.org/wsdl/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
 <types />
 <message name="nbButJoueur">
        <part name="nomJoueur" type="xsd:string" />
 </message>
 <message name=" nbButJoueurResponse">
        <part name="result" type="xsd:string" />
 </message>
 <portType name="ButJoueur">
        <operation name=" nbButJoueur " parameterOrder="nomJoueur ">
                <input message="tns: nbButJoueur" />
                <output message="tns:nbButJoueurResponse" />
        </operation>
 </portType>
```

Fig. 3 : Exemple de document WSDL

Le principal reproche que l'on fait aux documents WSDL est de ne décrire que les aspects syntaxiques des web services. Il n'y a pas d'information sur le sens des opérations et des paramètres. Nous verrons dès lors que certaines des méthodes de substitution que nous vous présenterons au chapitre 3 prennent le pli d'ajouter des informations sémantiques.

2.4.2 SOAP

SOAP est un protocole basé sur XML qui permet l'échange d'informations entre applications et l'appel de procédure à distance (Poiron 10). Les messages SOAP sont le plus souvent transportés via HTPP mais n'importe quel autre protocole de communication peut être utilisé.

C'est le protocole de transport qui permet de fournir au web service les paramètres d'entrée nécessaires et de récupérer les informations à valeur ajoutée promises par le WS.

L'utilisation des messages SOAP n'étant remise en cause par aucune des méthodes de substitution que nous verrons dans ce mémoire, nous n'entrerons dès lors pas plus dans les détails de son fonctionnement et de sa syntaxe.

2.4.3 UDDI

UDDI est un standard qui spécifie les protocoles pour l'accès à des registres de web services, des méthodes pour contrôler l'accès aux registres et un mécanisme pour distribuer ou déléguer des enregistrements aux autres registres.

Un registre UDDI fournit une approche standardisée pour localiser un web service, invoquer ce service et pour gérer des métadonnées à propos de ce service.

Les fournisseurs de services publient les interfaces des services qu'ils offrent dans un annuaire. Un client soumet ensuite à cet annuaire des requêtes de recherche de services. L'annuaire répond en envoyant l'adresse (URL[1]) du service correspondant. « Le client utilise ensuite cette URL pour l'envoi des messages traduisant l'invocation des opérations offertes par le service. » (Fauvet & al. 07)

La communication avec les registres UDDI se fait via l'échange de message SOAP.

Selon (Fauvet et al. 07) l'avenir des registres UDDI est incertain, d'une part en raison de leur intérêt limité aux aspects fonctionnels des web services, et, d'autre part, en raison de l'existence de moteurs de recherche.

[1] URL -**U**niform **R**esource **L**ocator–est utilisé pour adresser des documents sur le WWW (W3C 12).

2.5 Scénario d'utilisation

Maintenant que sont définies les notions intervenant dans l'utilisation des web services, voyons comment les employer pour parvenir à utiliser un web service.
Voici un schéma qui permettra de visualiser comment ces notions interagissent entre elles :

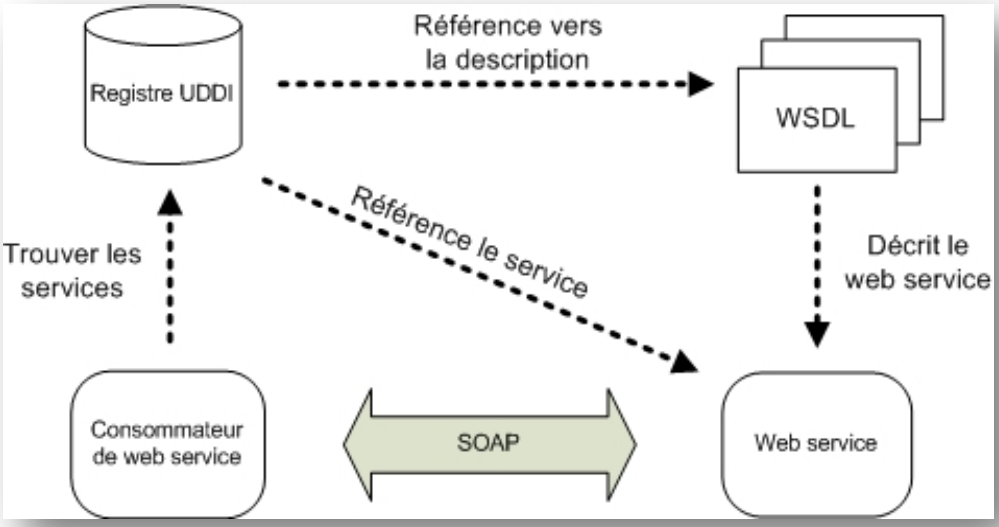


Fig. 4 : Eléments d'une plate-forme de web services

En partant du schéma de la figure 4, décrivons le cheminement classique pour parvenir à utiliser un web service :

Le web master (le consommateur de web service sur le schéma) souhaite trouver un web service qui lui calcule, pour deux équipes de division 1 données, laquelle a le plus souvent terminé dans les cinq premières du championnat.
Il interroge un registre UDDI pour trouver des web services qui pourraient correspondre à ses besoins. Cette étape se fait généralement de manière non-automatisée : le consommateur doit consulter lui-même le registre UDDI pour essayer de trouver les web services pouvant lui rendre service.
Le registre UDDI lui en présente un ou plusieurs qui correspondent aux critères qu'il a fourni. Pour s'assurer qu'ils lui conviennent bien, il récupère via le registre UDDI la référence du document WSDL descriptif du web service et des méthodes qu'il offre.
La lecture de ce document lui permet de mieux comprendre le web service grâce à sa description et lui permet de connaitre les arguments que les méthodes demandent pour effectuer le calcul ainsi que

les données qu'il va récupérer.
Il peut dès lors commencer à échanger des messages SOAP qui vont lui permettre d'établir la communication avec le web service, de lui fournir les paramètres d'entrée nécessaires à son utilisation et de récupérer les paramètres de sortie.

Comme vient de le faire le web master, l'utilisation classique d'un web service c'est d'y faire appel et d'obtenir la réponse à cet appel. C'est une utilisation « one shot ». Il peut cependant arriver que la notion d'état intervienne. C'est-à-dire que le résultat d'un appel à ce web service peut dépendre d'un appel précédent. Nous verrons plus loin quelle est l'implication de cette notion d'état dans la substitution de web services.

Nous avons à présent fait le tour des éléments intervenant dans l'architecture des web services et des interactions entre elles. Ces éléments sont les éléments fonctionnels, ceux qui permettent de découvrir, de décrire et d'utiliser un web service. Certaines méthodes de substitution que nous vous présenterons plus loin ne se basent que sur ces aspects fonctionnels. D'autres méthodes se basent non seulement sur ces aspects fonctionnels mais aussi sur des aspects non-fonctionnels. Les plus intéressants de ces aspects sont regroupés au sein de la notion de qualité. Voyons dans le chapitre suivant ce qu'est la qualité dans le cadre des web services et quels sont les aspects de qualité que l'on prend en compte.

2.7 La qualité dans les web services

Le choix d'un web service se fait d'une part sur base du service qu'il rend, ses fonctionnalités. Comme nous venons de le voir, c'est grâce au document WSDL qu'on sélectionnera le web service selon ce critère de fonctionnalité.

Mais ce choix peut également se faire sur base des caractéristiques non-fonctionnelles du WS, autrement dit, de sa qualité de service (**QoS** pour **Q**uality **O**f **S**ervice). Nous allons expliquer dans ce chapitre ce qu'est la qualité de service et quels sont les différents critères de qualité qui sont les plus souvent pris en compte.

La qualité de service permet de distinguer deux web services rendant les mêmes fonctionnalités. Elle permet au consommateur de web services de faire son choix entre deux web services de façon objective et non sur base d'une description plus attrayante ou la seule bonne réputation de son fournisseur de service.

Nous l'avons déjà évoqué dans l'introduction, un des critères de qualité nous intéresse particulièrement quand on parle de substitution : la **disponibilité**.

La disponibilité mesure à quel point un service web est présent ou prêt pour un usage immédiat (Huang & al. 09) (Mani & al. 06). La disponibilité représente la probabilité qu'un service soit prêt à répondre à une requête.

Dans le cadre de ce mémoire, nous allons accorder une attention particulière à la disponibilité du web service. Ou plutôt à sa non-disponibilité : que faire lorsque le web service dont on attend la réponse n'est pas disponible ? Attendre qu'il le soit de nouveau ou en trouver un autre qui, lui, pourra nous rendre le service demandé ?

La réponse tient dans ce mot : la substitution. Comme nous l'avons évoqué, le seul but de la substitution n'est pas de pallier la non-disponibilité d'un service.

On peut aussi vouloir substituer un service par un autre offrant de manière générale une meilleure qualité de service. Nous comprenons donc que malgré que la disponibilité en soit un des aspects, quand nous parlerons de qualité de service dans ce mémoire, ce sera pour parler des autres aspects de qualité. Voyons donc quels sont les autres critères de qualité les plus souvent pris en compte:

- **Le temps de réponse**

Le temps de réponse mesure le temps d'exécution (temps nécessaire à réaliser les fonctionnalités du service) et le temps d'attente (temps nécessaire aux autres activités. Le temps nécessaire à l'échange

de message par exemple) (Huang & al. 09) (Mani & al. 06). Le temps de réponse peut être un critère très important dans certaines utilisations du web service et n'être qu'un critère secondaire pour d'autres. La même remarque est valable pour différents web services utilisés par une même application. Notre site internet sur le championnat de Belgique accordera, par exemple, une grande importance au temps de réponse du web service lui renseignant en direct les scores des matchs du week-end mais il sera par contre beaucoup plus tolérant pour ceux lui offrant les statistiques du match, comme celui lui renseignant le nombre de passes maximales consécutives d'une équipe pendant un match.

- **La fiabilité**

La fiabilité fait référence à la capacité du web service à exécuter les fonctions requises, dans des conditions déterminées et un laps de temps spécifié. Cette capacité est généralement évaluée par le taux d'échec des demandes (par mois ou par an). (Lee & al. 03) (Shade & al. 12)

La fiabilité est souvent une notion importante à prendre en compte, quelle que soit l'utilité du web service. Une fois le travail de recherche et de sélection du WS correspondant le mieux à ses critères réalisé, l'utilisateur n'a pas envie de devoir recommencer en raison d'une faible fiabilité.

- **Le coût**

Le coût correspond au prix à payer pour effectuer une requête au service (Huang & al. 09) (Mani & al. 06). Cela peut être gratuit ou même onéreux dans le cadre de web services à haute valeur ajoutée.

Maintenant que sont vus les différents critères de qualité de service, avec un point d'attention particulier sur la disponibilité, nous pouvons entrer dans le vif du sujet : la substitution. Nous verrons ce qu'est la substitution, quelles sont les différentes sortes de substitution selon le point de vue duquel on se place (consommateur ou fournisseur du service) et quelles sont les réponses déjà proposées dans ce cadre.

Commençons donc par rappeler son utilité et décrivons succinctement les deux grandes familles de substitution : la substitution interne et la substitution externe.

3. La substitution de web services

Dans ce mémoire, nous avons pris le pli de classer les différentes sortes de substitution en deux grandes familles : la substitution interne et la substitution externe. Après avoir rappelé quelle est l'utilité de la substitution et quelles sont les notions transversales à tout type de substitution, nous allons donc vous donner une première explication de ces deux grandes familles.

3.1 Utilité et types de substitution

Comme nous l'avons déjà évoqué dans l'introduction, la substitution permet de faire face à trois situations :

- Le service utilisé n'est plus disponible.
- Le service utilisé n'offre plus la qualité de service attendue.
- On désire remplacer le service utilisé par un autre offrant une meilleure qualité de service.

Voici une mise en situation. Notre webmaster souhaite transmettre en temps réel les scores des matchs de foot du week-end. Plutôt que de devoir mettre à jour lui-même son site à chaque but, il met en place des appels à un web service spécialisé : goalForWk.

Le premier week-end de championnat, le serveur hébergeant le web service tombe en panne. Rien n'ayant été prévu pour pallier ce problème, les scores de ce week-end ne sont donc pas disponibles sur son site.

Ne souhaitant pas se mettre à dos les aficionados, le webmaster décide de mettre en place pour le week-end suivant une substitution de ce web service en cas de problème. Car, comme il le sait, « la substitution permet de faire face à une perte de disponibilité d'un web service » (Mani & al. 06).

Cette perte de disponibilité peut être, par exemple, due à un crash du serveur hébergeant, à une incapacité à répondre due à une surcharge temporaire ou encore simplement à une opération de maintenance planifiée. Le consommateur de service ayant, malgré tout besoin, de la fonctionnalité rendue par le web service, un substitut doit être trouvé.

Pour faire face à ce défi, deux grandes familles de substitution existent selon que l'on se place du point de vue du fournisseur de service ou du consommateur :

- La substitution interne (par réplication)

Dans le cadre de la substitution interne, on se place du point de vue du fournisseur de service qui veut assurer la disponibilité de ce dernier.

La substitution interne permet de pallier un des problèmes de la substitution : la perte de disponibilité d'une implémentation d'un web service utilisé.

Dans notre exemple, ce serait le fournisseur du service goalForWk qui, mis en garde par les nombreuses plaintes du week-end précédent, décide de fournir une solution pour qu'il soit toujours disponible.

Nous verrons dans le chapitre 3.3 ce qu'il doit mettre en place pour y parvenir et quels sont les choix qu'il a à poser.

- La substitution externe (lors de sélection)

Dans le cadre de la substitution externe, on sort du web service isolé dont on veut maintenir la disponibilité pour se placer dans un cadre plus général. Si un web service n'est pas disponible, on peut vouloir lui trouver un substitut qui répondra au service demandé avec un certain degré de qualité de service. Ce service sera, contrairement à la substitution interne, fourni par un autre fournisseur de service. Il s'agit de solutions pour améliorer la disponibilité du service lui-même.

De même, si un web service n'offre plus la qualité de service que l'on attend de lui, on peut vouloir le remplacer par un autre. Et même s'il continue d'offrir la même qualité de service, on peut vouloir, à tout moment, utiliser un service qui en offre une meilleure (Bravo & al. 10).

Le webmaster pourra, par exemple, faire appel au service butsDuChampionnat, jupilerLeagueGoal ou encore goalVanDeBelgisheKampionnat pour remplacer le service goalForWk auquel il faisait habituellement appel.

Nous verrons dans les chapitres 3.3 et 3.4 quelles solutions peuvent être apportées à ces problèmes et le cadre d'utilisation de chacune d'entre elles. Mais voyons d'abord une notion qui est transversale aux deux familles de substitution : l'aspect dynamique.

3.2 Substitution statique et dynamique

Un appel à un web service peut se faire de manière statique ou de manière dynamique (Tere & al. 12). L'appel statique suppose d'avoir construit les appels à une implémentation du web service lors du développement. Cela permet d'avoir des performances plus élevées lors de l'exécution mais offre moins de flexibilité par la suite dans le choix du web service. Le web service à utiliser est en effet défini une fois pour toutes lors du développement. L'appel dynamique suppose qu'on découvre et invoque le web service qui sera effectivement utilisé à l'exécution de notre

programme. En extrapolant ces définitions à la substitution, une substitution statique supposerait que les substituts sont trouvés lors du développement et mis en dur dans le code. Une substitution dynamique supposerait que les substituts sont trouvés et invoqués lors de l'exécution. Nous verrons plus loin dans ce mémoire différentes méthodes de substitution. Si on s'en tient à la définition que nous venons de voir, toutes ces méthodes peuvent être utilisées pour réaliser des découvertes et invocations dynamiques de web services. Dans certains cas, l'application cliente sera adaptée et il faudra accepter un délai de traitement allongé, délai qui découle de cette recherche dynamique.

Afin de pouvoir émettre des comparaisons entre les méthodes de substitution présentées, nous restreignons donc cette définition. Une méthode considérée comme proposant une substitution dynamique est une méthode qui laisse le client ignorer quel service (ou quelle implémentation du service) sera effectivement appelée à l'exécution.
Pour certaines méthodes, il peut avoir une influence sur ce choix en spécifiant la qualité qu'il souhaite voir atteinte mais si on parle de substitution dynamique, il ne pourra pas choisir lui-même quel web service ou quelle implémentation de ce web service sera effectivement appelé. Il fera son invocation à la méthode et se contentera ensuite de récupérer la réponse.

Nous pouvons à présent voir les deux grandes familles de substitution plus en profondeur, en commençant par la substitution interne et en poursuivant avec la substitution externe. Sachez déjà qu'un tableau de synthèse et une discussion reprendront les avantages et inconvénients de toutes les méthodes que nous verrons dans les chapitres suivants. Cela nous permettra d'avoir une vue d'ensemble des différentes méthodes de substitution.

3.3 Substitution interne (ou réplication de web services)

La substitution interne permet de faire face à une perte de disponibilité du web service.
Nous allons commencer par en décrire le principe pour ensuite décrire plus précisément les modules qui interviennent dans cette réplication.
Nous pourrons alors entrer dans les détails des caractéristiques propres aux réplications locales – toutes les implémentations du web service sont accessibles via une même adresse – et géographiquement réparties – les implémentations du web service sont réparties à travers tout le réseau – .

3.3.1 Principe de la substitution interne

L'idée de la substitution interne est que le fournisseur de service implémente son web service sur plusieurs web serveurs pour assurer sa disponibilité (un des critères de qualité du web service et une des raisons d'être de la substitution de web services). Ainsi, si l'un des sites tombe, les autres peuvent prendre le relais. On offre donc la même implémentation du service en remplacement. Il peut résulter une perte de la qualité de service en raison de la réplication (le serveur qui contiendra le réplica peut être moins performant ou plus éloigné du consommateur, d'où une augmentation du temps de réponse, un autre des critères de la qualité de service).

Selon (Osrael et al. 07), les travaux effectués jusqu'à présent empruntent beaucoup à la réplication d'objet dans les systèmes distribués. L'infrastructure est la même, la différence tient surtout dans l'échelle de la réplication : énormément d'objets dans le cadre de la réplication d'objets, peu de services dans le cadre de la réplication de service.

Décrivons en détail ce qu'un module de réplication doit contenir pour faire correctement le travail qu'on lui demande de faire, c'est-à-dire au moins ne pas envoyer de requête à un réplica qui ne serait plus disponible. Nous verrons ensuite que, selon que les réplicas sont disposés à un même endroit ou répartis à travers tout le réseau, les stratégies mises en place sont différentes et les bénéfices que l'ont peut en retirer ne sont pas toujours les mêmes.

3.3.2 Modules de réplication

Un service de réplication a des tâches plus variées que celle d'éviter d'envoyer une requête à un réplica qui ne serait plus disponible. Voyons quelles sont ces tâches et quels sont les modules permettant d'y faire face.

Six composants majeurs se trouvent dans la réplication de service, comme ils le sont dans la réplication d'objet (Osrael & al. 06) :

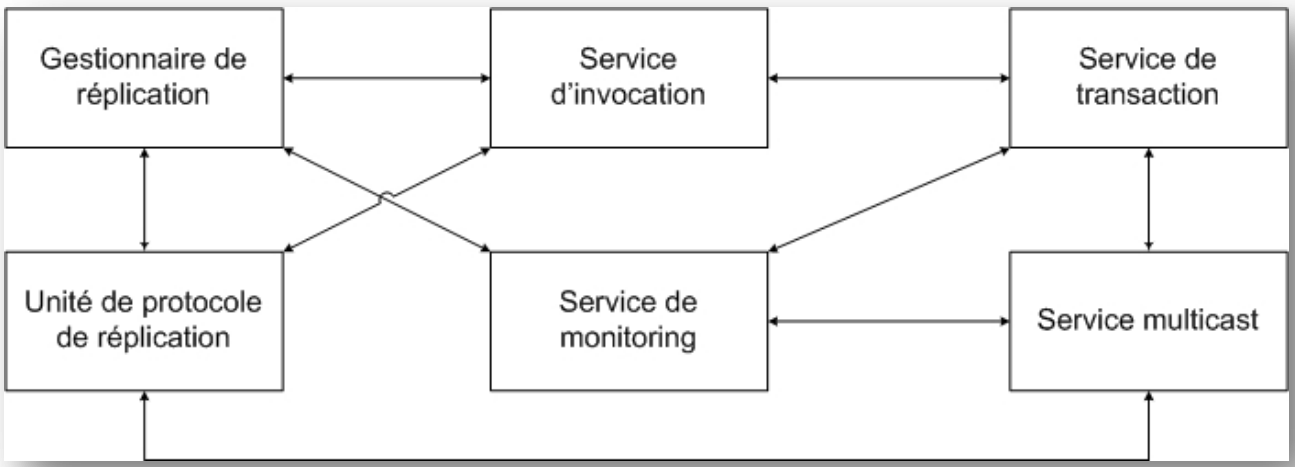


Fig. 5 : Composants d'un module de réplication

En pratique, tous ces services ne sont pas nécessairement implémentés mais ils peuvent l'être, selon les stratégies mises en place. Voici l'utilité de chacun d'entres eux, en prenant le parti de la réplication de service, car c'est elle qui nous intéresse dans le cadre de ce mémoire :

- Un **service multicast** pour un envoi fiable et ordonnancé des messages.

Les messages peuvent être envoyés à tous les réplicas ou à un seul réplica. Auquel cas, l'utilité du service multicast n'est pas prépondérante et le service d'invocation peut se charger du travail. Nous verrons plus loin que selon les niveaux de réplication (active –toutes les requêtes sont envoyées à tous les réplicas –, semi-active – seules les requêtes déterministes sont envoyées aux réplicas– ou passive – un seul serveur exécute les requêtes, les autres serveurs doivent être mis à jour –), un envoi ordonnancé des messages peut avoir son importance.

- Un **service de monitoring** pour surveiller les entités répliquées et pour la détection d'erreurs.

Il permet de prendre les actions appropriées en cas d'erreur : si un des réplicas du web service ne répond plus présent, le service de monitoring doit s'en rendre compte et le signaler pour qu'on puisse lui trouver un substitut et qu'on ne lui envoie plus de requêtes auxquelles il ne saura de toute façon pas répondre. Dans le même ordre d'idées, le service de monitoring doit aussi être capable de détecter qu'un réplica est de nouveau apte au service afin que des requêtes puissent à nouveau lui être transmises.

- Un **gestionnaire de réplication** pour la maintenance des services. Son but est de garder la trace de la localisation et des rôles des objets répliqués (maître ou backup).

Dans le cadre de réplication avec gestion de la cohérence (c'est-à-dire que toutes les requêtes sont envoyées à tous les réplicas), un mécanisme de synchronisation d'état doit être mis en place dans le cas où un nouveau réplica doit être ajouté au système.

Un moyen que l'on peut mettre en place est de stocker dans l'ordre toutes les invocations et de les rejouer sur le nouveau réplica. Cela peut être très consommateur de temps, surtout si le nombre de messages est grand.

Un autre moyen est d'employer un mécanisme de transfert d'état (via getting et setting), ce qui est certainement moins consommateur de ressources mais aussi moins simple à mettre en place.

- Une **unité de protocole de réplication** pour fournir la logique actuelle de réplication.

Cette unité peut s'occuper de réaliser la propagation des mises à jour vers les réplicas backup (dans une logique de réplication passive).

- Un **service d'invocation** pour fournir la logique d'invocation.

Intercepte les appels du consommateur et les transmet, selon la logique de réplication mise en place, à un ou à tous les réplicas.

Si la logique est d'envoyer une invocation à un réplica, le service d'invocation doit être capable de déterminer lequel est le moins sollicité au moment de l'invocation afin de répartir la charge de travail et d'optimiser les temps de réponse.

Si la logique est d'envoyer les messages à tous les réplicas, le service multicast se chargera de l'envoi.

- Un **service de transaction** optionnel pour la gestion des transactions entre entités répliquées.

Maintenant que nous avons passé en revue les modules qui doivent être mis en place pour gérer la réplication de service, penchons-nous sur la localisation des réplicas : locale ou géographiquement répartie. Chacune a ses avantages et inconvénients que nous allons vous exposer.

3.3.3 Localisation de réplication

Le choix de la localisation de la réplication a son importance quand il s'agit de mettre en place une stratégie interne de substitution de service. Nous allons détailler dans ce chapitre le fonctionnement de la réplication locale et géographique et quelles sont les spécificités de chacune en matière de gestion de la substitution.

Réplication locale (ou Servers' clustering)

Dans une stratégie de réplication locale, un ensemble de serveurs capables de répondre à la requête est disponible via une seule et même adresse.
Voici un schéma qui permettra de visualiser à quoi ressemble une réplication locale :

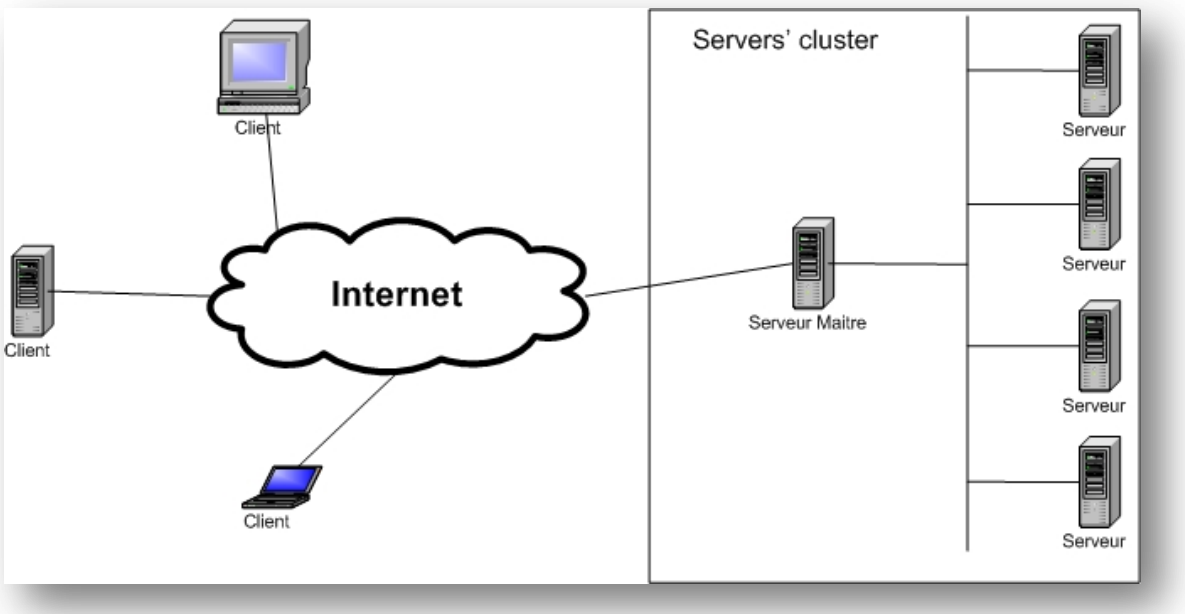


Fig. 6 : Architecture d'une réplication locale

Notons déjà que les requêtes arrivent sur un seul serveur, le maître, et qu'il répartit ces requêtes vers les autres serveurs. L'ensemble de ces serveurs est appelé cluster dans la suite du chapitre.
Reprenons notre exemple du webmaster et voyons quelle est sa confrontation avec un web service dont le fournisseur a utilisé une stratégie de substitution interne.
Ayant consulté les registres UDDI, le web master a trouvé le service butsDuChampionnat qu'il compte utiliser via son site internet. Il implémente dès lors des appels vers ce web service. Il se trouve que le fournisseur de ce web service a mis en place une stratégie de réplication interne. Le web master est au courant via le document WSDL où le fournisseur a eu la bonne idée de le préciser (dans la partie qui concerne la description du service). Le but est d'avoir un avantage concurrentiel par rapport aux fournisseurs du même genre de services qui n'auraient pas mis en place une telle stratégie de réplication interne.
Le webmaster n'a pas à se soucier de savoir lequel des réplicas contacter pour obtenir le résultat de sa requête : les serveurs ne sont pas visibles au delà du répartiteur (serveur noté Serveur Maître sur le schéma de la figure 6, chargé de la répartition des appels au web service). Il lance donc sa requête à l'adresse récupérée dans le document WSDL et n'a pas à se soucier d'autre chose. C'est au serveur maître à se charger de la répartition de ces requêtes.
Le webmaster ne saura pas quel serveur va effectivement lui répondre, il n'a même pas besoin d'avoir conscience qu'un ensemble de serveurs est capable de répondre à sa requête.

La requête du webmaster arrive donc au serveur maître, qui doit alors l'envoyer à un ou plusieurs des serveurs capables d'y répondre. Il y a, en effet, deux façons d'envisager la répartition des requêtes entre les serveurs selon que la cohérence entre eux est importante ou non. La cohérence entre les serveurs est importante si les services qu'ils hébergent ont des notions d'état. Elle n'est pas importante s'ils n'en n'ont pas. Comme on le suppose déjà, le fait d'avoir une cohérence à garder entre les serveurs implique des stratégies différentes à mettre en place.

Si la cohérence n'est pas un problème, chacun des serveurs contenant un réplica peut répondre à chacune des sollicitations venant de l'extérieur. Le serveur maître décide quel réplica va répondre selon la charge de chacun au moment de l'appel.
Tous les serveurs sont donc susceptibles de recevoir une requête arrivant dans le cluster. Le but poursuivi ici est de s'assurer qu'un des serveurs ne sera pas engorgé. Le module d'invocation que l'on a vu précédemment est en charge de s'assurer que ce but sera rempli.

Ayant été averti d'une défaillance éventuelle d'un des réplicas par le service de monitoring, le module d'invocation va substituer les appels au service défaillant par des appels à l'un des services

encore en place.

Concernant le service butsDuChampionnat utilisé par le web master, la cohérence entre les serveurs n'a pas été jugée par le fournisseur comme étant à prendre en compte. C'est donc une stratégie de réplication locale sans notion de cohérence qui a été mise en place. Le web master peut donc profiter de la garantie de disponibilité de son service tout en étant rassuré sur le fait que ses requêtes ne devraient pas rester bloquées dans un serveur engorgé.

Nous venons de voir que la cohérence entre les serveurs n'était pas toujours un problème. Elle peut cependant l'être. Il faut alors, à tout moment, que les serveurs aient le même état et répondent de la même façon aux requêtes.
Pour ce faire, il a trois manières de traiter les requêtes qui arrivent, trois niveaux de réplication : active, semi-active et passive.

- Si la réplication est active, toutes les requêtes sont transmises à tous les réplicas qui doivent les exécuter. Les différentes requêtes doivent être exécutées dans le même ordre par les différents réplicas pour qu'ils soient dans le même état.
Cela suppose que les réplicas soient déterministes : avec la même séquence de requêtes, ils produiront le même résultat.
Le fait qu'il y ait un ensemble de serveurs qui exécutent la requête doit être transparent pour le consommateur du service : hors de question que le web master reçoive cinq fois les résultats d'Anderlecht-Bruges s'il y a cinq serveurs qui traitent la demande. Le moyen d'y parvenir est de d'abord renvoyer toutes les requêtes vers le serveur maître. Selon la méthode mise en place, le client recevra sa réponse dès la première reçue par le serveur maître, après une majorité de réponses ou après toutes les réponses. La charge est laissée au serveur maître de ne renvoyer qu'une seule réponse au consommateur.
L'indisponibilité éventuelle d'un des serveurs ne pose ici aucun souci : si un des serveurs ne renvoie pas sa réponse, les autres le font et le client peut recevoir le retour de son appel.

- Si la réplication est semi-active, seules les requêtes déterministes sont répliquées. Les parties non déterministes sont exécutées par un serveur pointé comme le maître. Ce maître transmet les résultats de la partie non-déterministe aux autres serveurs.
En cas d'indisponibilité du serveur maître, un nouveau serveur est désigné comme maître par le gestionnaire de réplication. L'indisponibilité d'un des replicas ne pose pas plus de problème que dans la réplication active.

- Si la réplication est passive (appelée aussi approche primaire-backup), un serveur est désigné comme primaire, les autres sont backup de celui-ci.

Seul le primaire répond aux requêtes et transmet les mises à jour aux autres réplicas.

En cas de problème, un nouveau serveur est désigné comme primaire par le gestionnaire de réplication. Les mises à jour à effectuer depuis le serveur maître vers les réplicas peuvent être synchrones ou asynchrones.

Si elles sont synchrones, les mises à jour sont propagées aux réplicas avant que la réponse ne soit envoyée au client. Les réplicas sont donc toujours cohérents avec le serveur maître.

Si elles sont asynchrones, la réponse est envoyée au client dès que le serveur maître sait répondre. Les mises à jour sont propagées par la suite. Cela assure un meilleur temps de réponse mais les réplicas ne sont pas toujours cohérents avec le primaire.

Si un serveur backup devient indisponible, il n'y a pas d'intervention spécifique du protocole de réplication. Par contre, l'indisponibilité du serveur maître nécessite qu'un des serveurs de backup soit élu serveur maître.

Le problème principal de cette approche est que le serveur primaire reçoit toutes les requêtes et doit y répondre, il n'y a donc pas de répartition de la charge de travail, il peut s'en trouver engorgé et il en résultera une diminution de la qualité de service.

Le gros avantage de la réplication locale est que le consommateur du service n'a rien à faire pour profiter des avantages de la réplication qui lui assure la disponibilité de son service. La charge de travail se trouve du coté des serveurs. Si la cohérence entre les serveurs n'est pas importante, il peut résulter une amélioration de la qualité de service dû au fait de l'existence de ces réplicas : ce sera, en effet, le réplica le moins engorgé qui répondra à la requête. Les temps de réponse devraient donc en être améliorés. Si la cohérence entre serveurs est importante, les temps de réponse ne profiteront pas de cette réplication. Par contre la disponibilité du service est assurée.

Voyons à présent ce qu'est la réplication géographique, en quoi elle diffère de la réplication locale et quels sont les avantages que le consommateur peut en retirer.

Réplication géographique

Dans une stratégie de réplication géographique, les web services sont répliqués à travers tout l'internet en des localisations différentes.

Voici un schéma qui permettra de visualiser à quoi ressemble une réplication géographique :

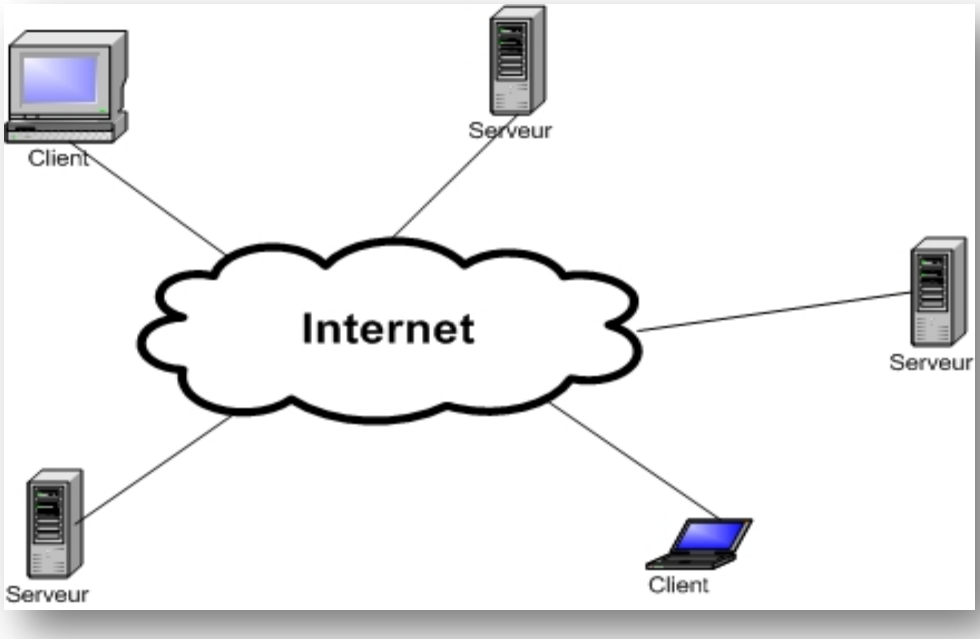


Fig. 7 : Architecture d'une réplication géographiquement répartie

Il n'y a plus un serveur central capable de répondre aux requêtes des utilisateurs comme dans la réplication locale mais bien un ensemble de ressources disponibles.

Cette technique permet de répartir l'accès au service à travers les réplicas avec le gros avantage que l'on peut choisir le réplica le plus proche du consommateur et dès lors de gagner le temps de propagation de la requête sur un site éloigné.

L'intérêt est de répartir la charge entre les différents serveurs mais aussi de se rapprocher du client et donc d'assurer des temps de réponse moindre. Deux qualités de service sont donc impactées par cette approche : le temps de réponse et la disponibilité.

Il y a deux manières d'envisager l'utilisation des réplicas.

Soit le client reçoit ses données de tous les serveurs. Dans ce cas le point d'attention se porte sur la manière de répartir les requêtes entre chaque serveur. On perd dans ce cas en partie l'intérêt de choisir un serveur proche du client. Mais on peut très bien imaginer une solution mixte : le client enverrait ses requêtes vers le serveur le plus proche quand il en a peu. Et répartirait ses requêtes sur

plusieurs serveurs s'il en a beaucoup. Ce qui permettrait de moins engorger un des serveurs.

Soit le client reçoit ses données d'un seul des réplicas. Dans ce cas l'attention se porte sur l'algorithme de sélection du serveur le plus approprié pour répondre au client.

Pour ce faire, il existe deux approches : l'approche du coté serveur et l'approche du coté client.

Si on opte pour l'approche du coté serveur, le choix d'un serveur se fait sans intervention du coté client mais via les serveurs, les routeurs ou les DNS.
Selon (Conti et al. 01), les solutions que l'on trouve ici sont des solutions propriétaires de produits commerciaux (Cisco et Nortel par exemple). Peu d'informations sont donc disponibles.

Il y a néanmoins un certain nombre de points communs qu'ils appliquent pour obtenir une certaine qualité de service :

- Un serveur spécifique est utilisé dans chacun des sites répliqués. Ce serveur peut agir comme serveur DNS ou rediriger les requêtes.
- Un site répliqué correspond à un ensemble de serveurs (cluster) et le serveur spécifique cité précédemment est utilisé comme passerelle entre cet ensemble de serveurs et Internet.
- Le serveur spécifique vérifie la disponibilité des serveurs du cluster par sondage.

Si on opte par contre pour l'approche du coté client, le choix du serveur se fait du coté du client.

Cette approche a comme grand avantage par rapport aux solutions du coté serveur que le consommateur a une vue d'ensemble du réseau.

Cela suppose que lorsque le client fait une requête DNS pour obtenir la résolution de l'adresse du web service qu'il a récupéré via le document WSDL, il obtient l'ensemble des adresses des réplicas. Contrairement à la réplication locale où le consommateur n'est pas au courant de l'existence d'une réplication (ou en tout cas, il n'est pas obligé de l'être), ici il a besoin de l'être. L'existence d'une réplication peut être d'ailleurs un argument de choix d'un web service plutôt qu'un autre.

Une fois que le client a obtenu la liste des réplicas, il peut envoyer une requête de test à chacun et classer les serveurs selon la rapidité de leurs réponses.

Si le réplica choisi venait à ne plus répondre aux requêtes, le consommateur peut le substituer par un autre de la liste. Ce qui entrainera sans doute une perte de qualité au niveau du temps de réponse puisque le réplica ne faisait pas l'objet du premier choix. Mais le consommateur sera assuré de la disponibilité du service, du moins tant que tous les réplicas ne tombent pas en même temps. Mais

cela est vrai de manière générale pour toutes les techniques de substitution.

Les différentes possibilités de substitution interne que nous avons vues sont utiles au consommateur de service qui est assuré de la disponibilité du service qu'il utilise. Les fournisseurs de service ont également intérêt à les mettre en place, pour leur éviter de perdre de l'argent en raison de l'indisponibilité de leur service ou pour leur permettre d'améliorer ou de garder une bonne réputation.

Mais, à la manière d'un portefeuille en bourse où la diversification est une bonne chose, il peut être intéressant de multiplier les fournisseurs de service. L'intérêt d'effectuer ce choix est multiple : on ne peut pas toujours être sûr qu'un fournisseur ait mis en place une (bonne) stratégie de réplication interne pour assurer la disponibilité de ses services. Il peut aussi très bien décider du jour au lendemain d'arrêter de fournir son service, d'augmenter ses prix ou encore de baisser la qualité offerte, le rendant moins intéressant d'un point de vue concurrentiel.

Ces risques, mieux vaut les éviter ou à tout le moins les réduire.

C'est l'objet d'intérêt des chapitres suivants : la substitution externe ou comment arriver à sélectionner un autre service capable de répondre à notre requête.

3.4 Substitution externe (ou substitution lors de sélection)

La substitution externe permet de faire face à une perte de disponibilité d'un web service en le remplaçant par un autre, proposé par un autre fournisseur de service, qui rend la même fonctionnalité. Commençons par en expliquer le principe pour ensuite décrire dans le détail différentes techniques de substitutions qui permettent soit de répondre à la perte de disponibilité, soit prennent en plus en compte les autres qualités de service.

3.4.1 Principe de la substitution externe

L'idée de la substitution externe est d'avoir une famille de web services substituable les uns par rapport aux autres.

Contrairement à la substitution interne, le but n'est pas d'offrir la même implémentation du web service répliqué sur différents serveurs mais d'avoir un ensemble de web services capable de répondre à une requête (plus ou moins efficacement).

L'intérêt peut être pour le consommateur de s'assurer qu'en cas de défaillance du service qu'il utilise, il peut en utiliser d'autres qui répondront à ses besoins.

La donne se complique si la qualité de service entre en ligne de compte. Il faut alors des stratégies pour d'une part connaître la qualité de service offerte, avoir des outils de mesures. Et d'autre part être capable de comparer et classifier les web services en fonction de ces critères.

Si tous les web services qui rendent la même fonctionnalité étaient proposés d'une façon standard, la substitution externe serait évidente : une recherche dans un annuaire UDDI selon un mot clé et on aurait tous les services substituts les uns des autres. L'utilisateur n'aurait plus, en cas de défaillance du web service utilisé, qu'à faire une recherche dans un annuaire et il trouverait un ou même plusieurs autres web services qu'il pourrait directement utiliser.

Nous devons apporter un bémol à cette vision idyllique.

Les web services proposés par les fournisseurs ne suivent pas de standard, ni dans leurs appellations, ni dans le nom des paramètres, ni dans leurs ordres, ni dans le type des paramètres d'entrée et de sortie. Pire, deux web services dont les fonctions portent le même nom et demandent les mêmes types de paramètres en entrée peuvent très bien proposer des fonctionnalités différentes. Toutes ces choses sont laissées à l'appréciation des développeurs de service, qui peuvent venir du monde entier, avec leur culture, leur sensibilité propre. Il ne faut donc pas attendre d'eux qu'ils aient exactement la même vision d'un problème.

Il peut cependant arriver que des web services soient parfaitement substituables : ils demandent les mêmes paramètres en entrée, donnent le même nombre de paramètre en sortie (et du même type) et

surtout pour un même set de paramètres d'entrée, les paramètres de sortie sont les mêmes.
Cependant, dans la plupart des cas, les types de paramètres en entrée sont différents, leur nombre peut l'être également. La même chose est vraie pour les paramètres de sortie.
Des stratégies doivent dès lors être mises en place pour, d'une part, détecter que les web services sont bien substituables entre eux, et, d'autre part, pour adapter les entrées-sorties afin de pouvoir utiliser un web service à la place d'un autre de manière transparente pour l'utilisateur, ou non transparente, selon les solutions.
Nous verrons que des algorithmes existent pour trouver les web services parfaitement substituables.
S'il s'agit d'aller plus loin et de trouver des WS qui sémantiquement rendent le même service mais qui diffèrent par leur syntaxe, nous verrons alors que la trilogie –SOAP, WSDL, UDDI – formant l'architecture actuelle des web services n'est plus suffisante.
Des nouveaux langages de description de service, plus riches que WSDL, sont proposés. De nouvelles infrastructures, plus complètes que l'UDDI sont nécessaires.

Nous commencerons par voir des solutions qui ne tiennent pas compte de la qualité de service. Puis nous verrons deux solutions qui la prennent en compte : la communauté de service et le Virtual web service.

3.4.2 La substitution externe sans prise en compte de la qualité de service

Ce chapitre nous permettra d'aborder des solutions de substitution qui permettent de faire face à une perte de disponibilité d'un web service sans que la prise en compte de la qualité ne soit un critère de sélection. Après en avoir rappelé le principe général, nous allons vous présenter quelques solutions intéressantes.

3.4.2.1 Principe

Votre application utilise habituellement un certain web service et vous n'avez pas besoin de prendre en compte la qualité de service pour la fonctionnalité qu'il vous apporte. Ce service devient tout à coup indisponible. Une opération de maintenance, une surcharge du serveur, une relocalisation, les raisons ne manquent pas. Mais que faire pour faire face à ce problème? Parcourir manuellement les registres UDDI pour trouver un substitut en comparant les documents WSDL récupéré?
Voyons plutôt dans ce chapitre les solutions apportées par quelques auteurs pour trouver un substitut

au web service injoignable de façon plus automatique. Ce type de substitution a en commun avec la substitution interne de permettre de pallier le problème de la disponibilité mais pas le problème de qualité, ce qui peut être utile dans les cas où la prise en compte de la qualité n'est pas nécessaire.

Voici une mise en situation pour mieux cerner la problématique de la substitution externe.
Afin d'offrir toujours plus de contenu sur son site, le web master propose depuis peu les résultats de 3ème provinciale récupérés via le web service mis à disposition par la fédération. Il sait qu'il ne peut attendre de qualité de service, les budgets étant très serrés. Il ne veut cependant pas décevoir les visiteurs de son site qui pourraient être privés, en cas de défaillance du service, des résultats le week-end où se joue le titre et il veut donc mettre en place une substitution de ce service.
Il aurait certes pu mettre en place une substitution statique en trouvant lui-même les web services permettant de récupérer la même information. Il aurait alors modifié son application pour que si le web service ne répond pas, soit appelé un autre web service.
Mais le même travail de recherche aurait du être effectué pour la deuxième provinciale, la première, la division 3a, 3b, …
La tache lui semblant trop importante, une autre technique doit être trouvée.

Pour l'aider, les deux premières solutions que nous vous proposons sont appelées à être déployées du côté du client ou à être implémentées comme des web services à part entière. Ces deux solutions permettent de trouver un web service qui pourra être utilisé comme substitut d'un autre. Mais elles supposent encore de mettre en place du coté du client une détection de la défaillance du service et le remplacement des appels à ce service grâce à son substitut. Les deux suivantes demandent la mise en place de toute une infrastructure avec toute la gestion que cela implique : qui se charge de maintenir cette infrastructure, de garantir son fonctionnement et son efficacité ?
Détaillons ces solutions, la façon dont elles fonctionnent et donnons-en à chaque fois une première appréciation.

3.4.2.2 Substitution externe via recherche de substitut automatique

Si la substitution interne n'a plus beaucoup de secret pour lui, le web master ne s'y connait pas encore en substitution externe. Intuitivement, il se dit cependant que si les paramètres d'entrée de deux web services sont de même nombre et de même type, il y a déjà une chance pour que les services offerts correspondent. Si en plus, les paramètres de sortie sont aussi de même type, les chances augmentent encore. Reste alors à essayer ces deux web services : si, à mêmes paramètres d'entrée, les résultats retournés par les web services sont les mêmes : c'est gagné, ils peuvent être utilisés comme substituts. Cette méthode ne résoud pas tout le problème de la substitution : il faut encore arriver à détecter l'échec de l'appel au premier web service et remplacer cet appel par le substitut trouvé. Elle permet cependant d'apporter une première réponse au défi le plus important, c'est-à-dire trouver deux web services qui peuvent être utilisés comme substitut l'un de l'autre.

Ernst et al

Cette façon de détecter si deux web services sont substituts l'un de l'autre, (Ernst et al. 06) y ont pensé avant lui. Leur idée est d'avoir conçu un algorithme de détection automatique des substituts envisageables pour un web service.

Dans sa première phase, l'algorithme regarde si les paramètres d'entrée des web service potentiellement substituts couvrent les paramètres d'entrée du web service substituable. On opère par là une première sélection. Que les paramètres d'entrée correspondent est une condition nécessaire dans cette conception de la substitution mais certainement pas suffisante : rien ne garantit à ce stade que les web services vont renvoyer la même information, tout juste qu'il n'y a pas de transformation nécessaire pour utiliser l'un à la place de l'autre.

L'algorithme vérifie donc ensuite que les paramètres de sortie correspondent entre eux. Un set d'exécution des web services est alors observé. Si pour les mêmes paramètres d'entrée, deux web services renvoient systématiquement des paramètres de sortie qui correspondent, c'est qu'ils sont substituables.

Les auteurs ont testé leur algorithme sur une série de web services. Ils n'ont pas constaté de faux positifs ni de faux négatifs.

L'idée est intéressante pour trouver des web services qui correspondent parfaitement mais dans la réalité, deux web services sont rarement totalement identiques. Une adaptation est souvent nécessaire pour les faire correspondre et à ce stade, l'algorithme ne permet pas une telle approche : un web service qui renverrait le nombre de but marqué en moyenne par un joueur sur un match

comme une chaine de caractère « 0.3 » et un autre comme un nombre 0.3 seraient par exemple considérés comme non substitut l'un de l'autre alors que dans la réalité ils le sont, à une transformation près. On peut imaginer des adaptations de cette méthode qui pallieraient ce problème.
Pour profiter de cette méthode, il faut déployer la solution du coté du client et rechercher les substituts avant que la non-disponibilité survienne. La recherche d'un substitut demande en effet d'observer un set d'exécution des différents web services, ce qui n'est pas possible si le web service à substituer n'est plus disponible. Il s'agit donc d'une substitution statique : il faut rechercher les substituts à l'avance et en trouver suffisamment pour qu'on sache en contacter au moins un en cas de problème.

Le web master est déjà content de sa découverte. Mais il se dit que, si faire une recherche sur les types de paramètres peut se révéler gagnante, pourquoi ne pas faire une recherche sur les noms des paramètres ?
La solution de (Dong & al. 04) pourrait dès lors lui convenir.

Woogle

Partant du constat qu'une recherche de web services dans un registre UDDI en se basant sur des mots clés peut se révéler infructueuse et nous faire passer à côté d'un web service qui nous aurait parfaitement convenu, (Dong & al. 04) ont eux aussi imaginé un algorithme qui permet le remplacement d'un web service par un autre.
Leur algorithme, woogle, permet la recherche d'un web service qui rend des fonctions similaires à un autre, que l'on sait nous convenir. Comme l'algorithme proposé par Ernst & al., woogle commence par vérifier si les opérations des différents web services ont des paramètres d'entrée similaires.
Cet algorithme se base pour cela sur les noms des paramètres plutôt que sur leur type, comme le faisait le précédent.
Comme cela peut être très restrictif car soumis au bon vouloir du développeur du web service, les auteurs ont imaginé des algorithmes de recherche élaborés en se basant sur l'heuristique selon laquelle « deux paramètres tendent à exprimer le même concept s'ils apparaissent souvent ensemble ». Leur constat est le suivant : les noms de paramètre de fonction sont souvent des concaténations de mots dont la première lettre est notée en majuscule. L'algorithme classe donc les mots selon leurs co-occurrences. Il décrète alors que si un mot apparaît sans l'autre, il y a de forte chance que son sens soit le même que si les deux apparaissent ensemble.

Il parcoure également les descriptions textuelles des web services présentes dans les documents WSDL pour y découvrir des similarités, de la même façon qu'il le fait pour les paramètres des fonctions.

Un des problèmes que l'on peut citer concernant cette méthode est que si la recherche du substitut est automatique, le remplacement du web service à substituer ne l'est pas : non seulement une personne physique doit intervenir pour vérifier la compatibilité entre les web services mais en plus, un matching entre les paramètres doit être effectué pour qu'ils correspondent puisqu'ils ne sont potentiellement pas du même type. De plus, comme pour l'algorithme précédent, trouver un substitut doit se faire en amont de l'indisponibilité effective du web service à substituer. Ce qui rend moins efficace la substitution : il n'y a aucune garantie que le substitut trouvé sera disponible au moment où on en aura besoin.

Comme nous le voyons, les algorithmes de recherche automatique peuvent être utiles dans certains cas relativement simples de substitution : ceux où, soit les types des paramètres, soit les noms de ces paramètres correspondent, selon la méthode utilisée. De plus, si ces méthodes permettent de trouver des substituts, elles font l'impasse sur la détection de l'échec du service à substituer et son remplacement. C'est donc à la charge du client de mettre en place des stratégies pour répondre à ces défis.
Il faut des solutions qui ne se limitent plus à un algorithme de recherche mais mettent en place toute une infrastructure de classification des web services, de détection d'échec, de recherche de substitut et de remplacement du web service à substituer.
Nous allons dès lors vous présenter dans le chapitre suivant deux méthodes qui permettent des recherches de substitut plus élaborées et qui prennent en charge la substitution de service dans son ensemble.

3.4.2.3 Substitution externe via service de substitution dynamique

Les deux propositions que nous allons voir maintenant, les auteurs les ont imaginées dans le cadre de composition de service. La composition de service permet d'obtenir une nouvelle fonctionnalité en combinant plusieurs web services.
Si le cadre est la composition de service, le mécanisme de substitution mis en place n'est pas propre à la composition et peut très bien être pris dans un cadre plus général. Les deux méthodes que nous allons vous présenter ont quelques points communs : elles se passent de registre UDDI en maintenant leurs propres registres, ceux-ci qui contiennent des informations sémantiques sur les web services. Elles ont chacune un service responsable de mettre dynamiquement en place la substitution. C'est-à-dire qu'une fois que le consommateur a effectué sa requête, il ne doit plus se soucier de savoir quel service va lui répondre et si une substitution sera effectuée ou non.
Les deux méthodes diffèrent grandement entre elles dans la façon de classer les web services, de trouver le substitut et de valider la substitution. Voyons tout cela dans le détail.

Substitution externe via district web virtuel

Selon (De Antonellis & al. 03a et 03b) un district virtuel est défini comme étant un consortium de membres indépendants qui coopèrent dans le but d'exploiter des opportunités d'affaires. Autrement dit, faire de la composition de service.
Dans le but de faire coopérer les membres d'un district virtuel, ils ont imaginé et implémenté une infrastructure : VISPO, pour **V**irtual district **I**nternet-based **S**ervice **P**latf**O**rm. Cette infrastructure gère l'exécution des processus coopératifs entre les membres du district. Dans le chapitre sur les communautés de service, nous verrons que le concept de communauté regroupe des membres qui offrent le même type de services. Le concept de district regroupe, quant à lui, des membres qui ont pour but de coopérer ensemble pour réaliser des compositions de services.
Avant d'aller plus loin dans la description du mode de fonctionnement de cette architecture, voyons quels sont les acteurs et les modules qui en font partie.

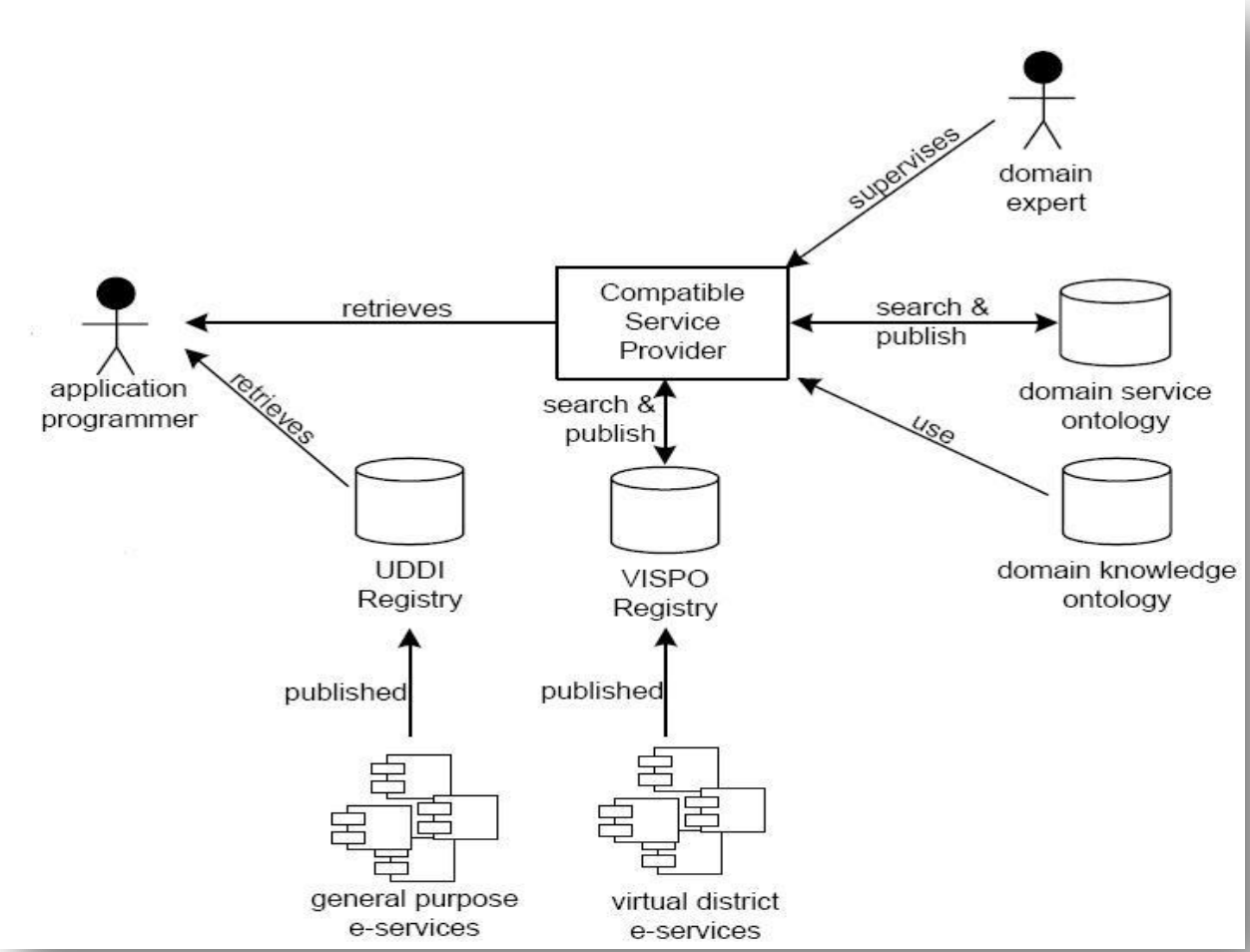


Fig. 8 : Eléments d'une plate-forme VISPO (De Antonellis & al. 03a)

- Le **C**ompatible **S**ervice **P**rovider : module de l'infrastructure VISPO en charge de la publication de service. Il est également en charge de la prise des actions nécessaires concernant la substitution : via une spécification de service, il est capable de trouver un service compatible qui peut être utilisé pour la substitution.

- Le **D**omain **K**nowledge **O**ntology : les termes utilisés dans un contexte particulier, un domaine, sont enregistrés au sein de cette base de données. Cette base est peuplée par le Domain Expert.

- Le **D**omain **S**ervice **O**ntology : organise les web services selon leurs relations sémantiques. Les services sont organisés dans le Domain Service Ontology selon les relations *is-a* – l'un des services est substitut de l'autre mais l'inverse n'est pas vrai –, *equivalent-to* – les deux services sont parfaitement substituables –, *similar-to* – les deux services sont semblables

mais une adapation pour utiliser l'un à la place de l'autre doit être réalisée –.

- Le **D**omain **E**xpert : spécifie les pré et post conditions des web services afin de mieux définir le comportement des web services. Il remplit les Domain Knowledge Ontology.

- Le **V**ispo **R**egistry : contient les documents WSDL décrivant les web services présents au sein des districts virtuels, à la façon d'un registre UDDI. De plus, un document descriptif complémentaire est généré : le Service Descriptor. Ce document va faciliter la découverte de web services compatibles.

Voyons à présent comment ces différents concepts vont interagir entre eux pour

a) Permettre l'ajout d'un web service au sein de l'infrastructure VISPO.
b) Permettre la réalisation de la composition de service.
c) Effectuer la substitution de service.

a) Ajout d'un web service

Lorsqu'un fournisseur veut qu'un de ses services fasse partie d'un district virtuel, il s'adresse au Compatible Service Provider en lui fournissant le document WSDL descriptif du service. Le web service est alors ajouté dans la base de données Vispo Registry, avec une définition des pré et post-conditions du web service si elle est disponible. Une classification de ce web service est alors réalisée via une analyse de compatibilité avec les autres services présents dans le registre.
Cette analyse de compatibilité permet de construire le Domain Service Ontology et est effectuée sous la supervision du Domain Expert.

Voici quelles en sont les étapes. A noter que ces étapes sont effectuées à nouveau lors de la recherche des web services compatibles avec un service abstrait devant intervenir dans une composition de service.
Deux phases sont définies pour classer le web service entrant dans l'architecture VISPO:

- **L'analyse de haut niveau** utilise les descriptions contenues dans les documents WSDL et les informations sémantiques ajoutées par le Domain Expert (les pré et post condition) pour trouver l'ensemble des web services compatibles avec la définition abstraite définie plus haut.
- **L'analyse structurelle :** pour chacun des web services sélectionnés lors de la phase 1, les messages et données à échanger sont analysés.

Comme l'algorithme Woogle que nous avons vu précédemment, l'infrastructure VISPO se base sur

le nom des arguments d'entrée et de sortie pour déterminer si deux web services sont compatibles ou non. Les synonymes, hyperonymes (terme dont le sens inclut le sens d'autres termes (Le Petit Larousse 04)) sont repérés, tout en prenant en compte le contexte dans lequel s'exécute la composition.

La même démarche est effectuée pour la description des méthodes.

A l'issue de ces deux phases d'analyse, une classification peut être déterminée entre les web services. Cette classification est sauvée au sein de du Domain Service Ontology au moyen des opérateurs equivalent-to, is-a et similar-to.

S_i equivalent-to S_j indique que les deux web service sont parfaitement substituables l'un par rapport à l'autre, il n'y a aucune adaptation à effectuer pour utiliser un web service plutôt qu'un autre.

S_i is-a S_j indique que le service S_i peut se substituer au service S_j mais que l'inverse n'est pas vrai.

S_i similar-to S_j indique que les deux web services ont un certain degré d'affinité mais qu'une vérification par le Domain Expert s'impose et qu'il faudra un effectuer un mapping pour substituer l'un à l'autre.

b) Composition de services au sein de l'architecture Vispo

Les web services qui vont réellement intervenir dans la composition de services ne sont pas définis dès le départ. Mais une description abstraite des différents services devant intervenir dans la composition est réalisée. Un document WSDL décrivant chacun des services abstraits est défini.

Juste avant l'exécution de la composition, les web services devant réellement intervenir sont recherchés. Cette recherche se fait de la même façon que si la description abstraite devait être ajoutée au sein de l'architecture Vispo. Les étapes que nous avons vues au point a) sont donc rejouées.

Un certain degré d'affinité est retiré des deux comparaisons et les web services sont classés par ordre croissant d'affinité avec la description abstraite du web service devant intervenir dans la composition et rangés dans ce que les auteurs ont appelé une classe de compatibilité. Elle regroupe les web services qui sont considérés comme substitut les uns des autres après la phase d'analyse de haut niveau. Un mapping éventuel peut devoir être effectué entre les différents membres de cette classe de compatibilité. Cette classe de compatibilité peut être mise en lien avec la Communauté de l'approche par communauté de service : les deux regroupent des web services rendant le même service et pouvant être utilisés comme substitut les uns des autres.

L'analyse structurelle peut alors commencer. Cette phase permet de faire le mapping entre les paramètres d'entrée et de sortie de la définition abstraite et ceux des web services sélectionnés.

Après cette phase de recherche, vient l'exécution de la composition. Un des web services devant intervenir peut se révéler indisponible. Il faut dès lors lui trouver un substitut parmi d'autres web

services compatibles et définir les règles de mapping nécessaire pour les faire correspondre.
Après ces deux phases, les web services compatibles entre eux sont trouvés et classés hiérarchiquement et les mappings nécessaires sont définis avec la description abstraite du service.

c) Substitution de service

Si un des web services sélectionné dans la phase précédente n'est plus disponible au moment de l'exécution de la composition, un substitut doit lui être trouvé. Grâce au classement effectué dans le point a), il suffit de regarder les liens au sein du Domain Service Ontology :
Si au moins un service est trouvé avec un lien is-a ou equivalent-to, la substitution est automatique et peut uniquement se baser sur le Domain Service Ontology. Cette substitution doit être validée par le Domain Expert.
Si similar-to : un effort de mapping doit être effectué par le Domain Expert. Ce mapping est sauvegardé au sein de la classe de compatibilité.
Si aucun lien n'est trouvé, les étapes du point b) sont recommencées sur les services disponibles dans des registres UDDI classique, ce qui peut se révéler beaucoup plus fastidieux.

La substitution au sein de district virtuel fait pour le moment l'impasse sur la qualité de service. Les auteurs affirment cependant que cela fera l'objet d'un futur travail.
Un autre problème peut être la nécessité d'avoir un Domain Expert dont le travail est indispensable à la substitution : non seulement il est l'acteur habilité à écrire les pré et post conditions si elles ne sont pas disponibles et donc à classifier les services selon leurs liens. Mais en plus il doit valider la substitution dans le cadre de web services détectés comme parfaitement substituables et écrire les règles de mapping en cas de services non-parfaitement substituables, contrairement à la substitution via service web virtuel où les règles de mapping sont réalisées par le fournisseur du service. Cela parait être en contradiction avec la définition des web services que nous avons vue au début de ce mémoire : « Il s'agit d'un ensemble de fonctionnalités exposées sur internet ou sur un intranet, par et pour des applications ou machines, sans intervention humaine ». Telle que conçue, l'infrastructure VISPO demande au contraire une intervention humaine régulière et importante.

Substitution externe via Siroco

(Fredj et al. 08) proposent une infrastructure middleware qui permet la reconfiguration d'une composition suite à l'indisponibilité d'un des services et ils l'ont nommée SIROCO. Partant du constat qu'à tout moment, un service utilisé dans une composition pouvait devenir indisponible, ils ont conçu une infrastructure middleware qui permet la substitution dynamique de ce service en prenant en compte sa sémantique.

Selon les auteurs un processus de reconfiguration tient en deux phases :

Phase 1) Découvrir la liste des substituts possibles parmi une liste de services compatibles sémantiquement

Phase 2) Identifier parmi eux lequel peut être utilisé comme substitut actuel.

Pour réaliser ces deux phases, l'infrastructure SIROCO offre :

- Un registre de service contenant les informations sur les web services connu du middleware : le ServiceRegistry. Il n'est donc plus ici question de se baser sur les registres UDDI.
- Une unité de gestion et d'exécution des compositions : l'ExecutionEngine.
- Une unité de monitoring en charge de vérifier que les compositions se passent correctement : le MonitoringManager.
- Et le module qui nous intéresse le plus, un adaptateur qui est en charge de reconfigurer dynamiquement les compositions quand c'est nécessaire : l'AdaptationManager.

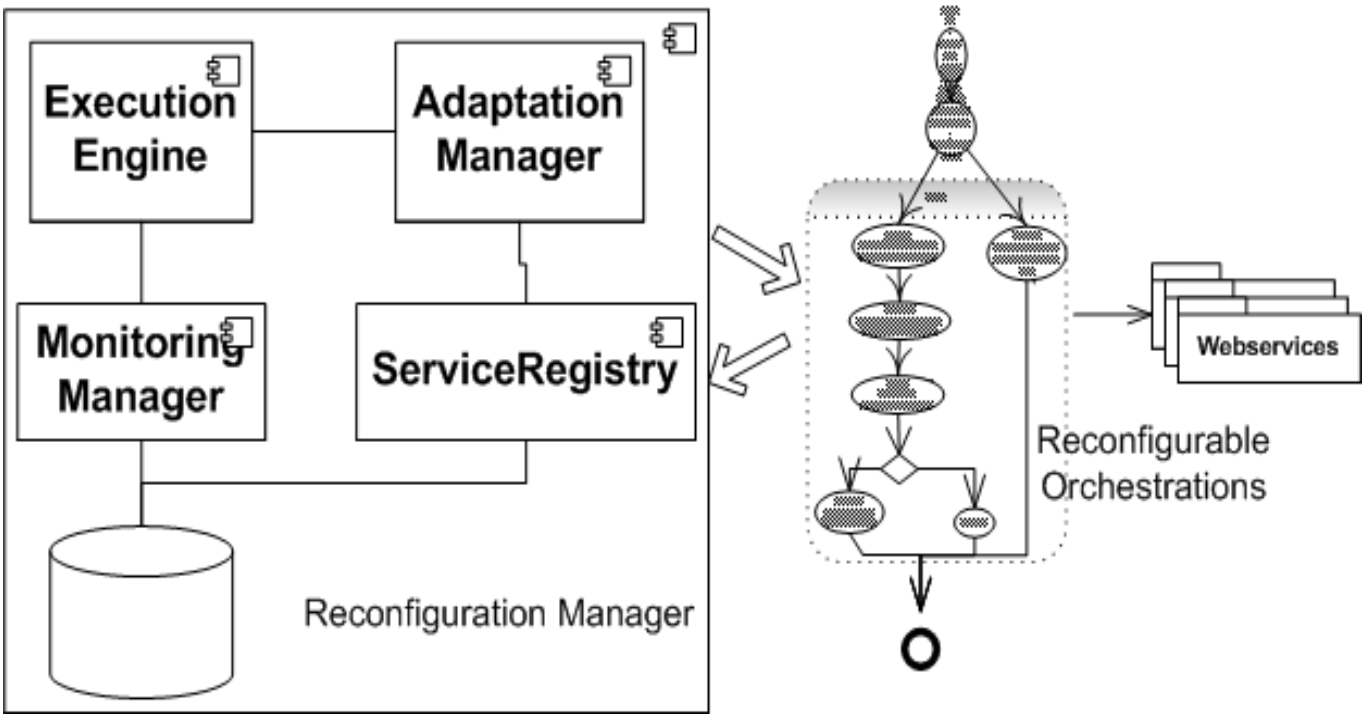


Fig. 9 : Eléments d'une platforme SIROCO (Fredj & al. 08)

Le serviceRegistry maintient un ensemble de catalogues de services. Chaque catalogue est caractérisé par une charte à laquelle les services réellement implémentés doivent satisfaire pour faire partie du catalogue. Lorsqu'un service est ajouté dans un des catalogues de la plate-forme, il fournit un document SAWSDL : il s'agit d'un document WSDL agrémenté d'annotations sémantiques. Ce document est donné par le fournisseur et permet de faire le lien entre la charte imposée par le catalogue de service et les services qui se trouvent dans le catalogue. Un autre document que le fournisseur peut, mais ne doit pas, fournir est un document descripteur des états du service. Celui-ci sera utilisé pour rechercher un autre web service dont le document est compatible. C'est-à-dire qu'on peut faire correspondre les états du service à substituer et de son substitut.

En cas de problème lors d'une composition, l'unité de monitoring le détecte et en avertit le module de reconfiguration. L'AdaptationManager recherche alors toutes les compositions potentiellement impactées par le web service problématique et bloque ces compositions pour éviter toute nouvelle erreur. Une fois qu'il s'est assuré qu'une autre composition ne peut pas être impactée par le service défaillant, l'AdaptationManager peut se mettre à la recherche d'un substitut pour ce WS, ce qui est grandement facilité par le travail effectué en amont dans le ServiceRegistry.
Celui-ci est dès lors sollicité pour renvoyer la liste de ces potentiels substituts. Si le web service à

substituer ne possède pas de document descripteur de son état, il suffit au ServiceRegistry d'aller chercher dans le catalogue correspondant à ce WS et de renvoyer la liste des membres. Si le web service en possède un, il faut en plus trouver un autre web service dont le document descripteur correspond et synchroniser leurs états.
L'AdaptationManager en sélectionne alors un et le contacte pour vérifier sa disponibilité.
S'il est disponible, le substitut est trouvé et il traitera dès lors la demande. S'il est également indisponible, on passe au suivant.

La substitution est ici dynamique : c'est à l'invocation d'un web service que le MonitoringManager détecte le problème et averti l'AdaptationManager pour qu'un substitut lui soit trouvé. Le principal désavantage que l'on pourrait citer par rapport à une substitution mise en place statiquement c'est le temps de réponse qui s'en trouve rallongé. Il faut en effet compter la détection de l'erreur, la prise en charge par le module de reconfiguration, la recherche dans les registres des web services compatibles, la vérification de leurs disponibilités. Tout cela entrainera une baisse de la qualité en raison de l'allongement du temps de réponse. Par contre, on est assuré de contacter un web service qui est disponible au moment de l'invocation, ce qui n'est pas négligeable.
Le fait que l'état du web service à substituer soit pris en compte lors de la substitution est un grand avantage de cette méthode.

La description de la méthode SIROCO clôture notre tour des principes de substitution externe sans prise en compte de la qualité. S'il faut tenir compte de la qualité, le composant en charge de trouver le substitut voit sa tâche compliquée. Intéressons-nous, dans les deux propositions suivantes, à la façon dont il arrive à surmonter cette difficulté.

3.4.3 La substitution externe avec prise en compte de la qualité de service

Dans ce chapitre, nous allons aborder des méthodes de substitution qui permettent non seulement de répondre à une perte de disponibilité d'un web service mais permettent aussi de prendre en compte les autres aspects des qualités de service.
Après en avoir expliqué le principe, nous allons voir quelques méthodes intéressantes.

3.4.3.1 Principe

La prise en compte de la qualité dans le cadre de la substitution complique encore un peu plus le travail. Il n'est alors plus seulement question de comparer des services web sur base de leurs

caractéristiques fonctionnelles, il faut de plus voir lequel des services correspond le mieux aux besoins du consommateur du web service, ou, de manière plus générale, lequel offre la meilleure qualité de service.
Une gestion de la qualité est dès lors nécessaire avec toutes les questions que cela engendre : quelle est la qualité offerte par ce web service, comment l'obtient-on et sur lesquels des critères de qualité base-t-on la comparaison.

3.4.3.2 Substitution externe via service de substitution avec prise en compte de la qualité

Nous allons voir que si le principe de base des deux solutions que nous vous proposons ne diffèrent pas tellement, la mise en œuvre bien. Et la gestion de la prise en compte de la qualité est très différente.

Substitution externe via service web virtuel

(Vilas & al. 04a et 04b) ont défini le service web virtuel (VWS pour Virtual Web Service) comme étant un service publié de la même façon qu'un service classique et pouvant être appelé comme tel. Il n'y a aucune différence du point de vue du client entre un service réel et virtuel.
Le client pourra donc découvrir et faire appel à un web service virtuel comme s'il s'agissait d'un web service classique. Voyons dès lors quels avantages il peut tirer d'un VWS.
Ce service virtuel regroupe un ou plusieurs services qui seront appelés du côté du serveur, de façon transparente pour le consommateur. Le service virtuel agit comme un intermédiaire qui est invoqué par le client. D'un appel classique direct, l'on passe donc à un appel indirect.
Dans ce cadre trois acteurs sont définis, comme nous le voyons dans la figure suivante :

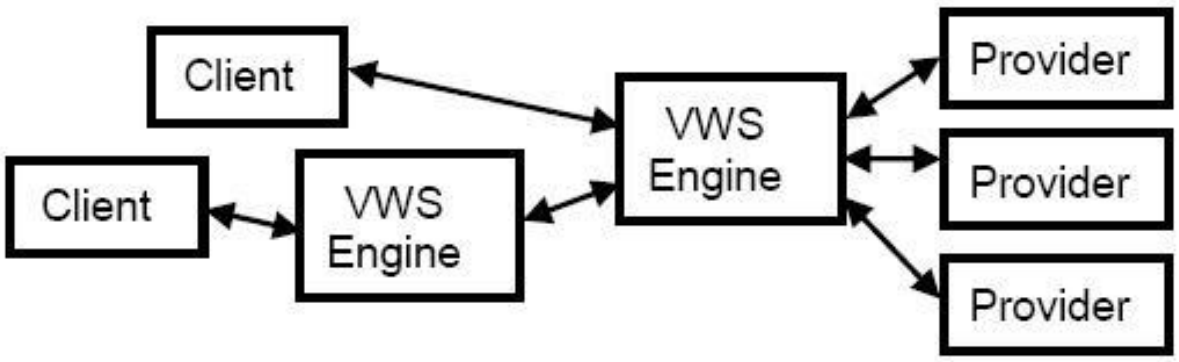


Fig. 10 : Eléments d'une plate-forme de service web virtuel (Vilas & al. 04a)

- Le client est l'acteur qui a besoin de la fonctionnalité offerte par un web service et l'invoque.
- Le fournisseur est l'acteur qui fournit le web service et le publie (noté Provider dans le schéma de la Fig. 10).
- Le web service virtuel qui regroupe un ensemble de web services (noté VWS Engine dans le schéma de la Fig. 10).

Pour qu'un web service traditionnel trouve sa place au sein d'un web service virtuel, il faut qu'un document qui décrit les fonctionnalités offertes par ce VWS soit consultable.
Les auteurs ont pour se faire imaginé un nouveau langage : le VWSDL pour Virtual Web Services Description Langage. Ce langage est basé sur XML, comme l'est l'architecture traditionnelle des WS, et son but est de décrire le web service.
Son utilité est donc la même que le langage WSDL mais sa structure est différente. En plus de WSDL, le VWSDL permet aussi de faire le mapping entre le VWS et les WS qui seront réellement appelés.
Voyons plus en détail la structure (ici légèrement simplifiée) du virtual web service description langage.

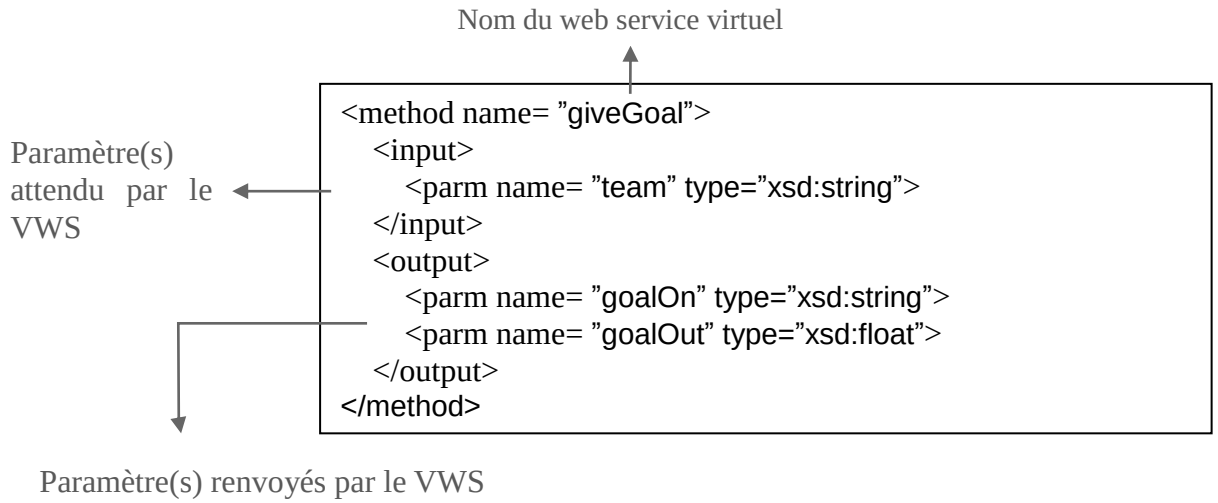


Fig. 11 : Eléments descriptifs d'un VWS

Grâce à cette figure, nous pouvons voir comment est défini un VWS : un nom, des paramètres d'entrée et des paramètres de sortie. Rien de bien neuf jusqu'à présent.
Pour faire le lien entre cette définition et le WS implémenté, il convient d'ajouter un nœud « invoke » dans le document :

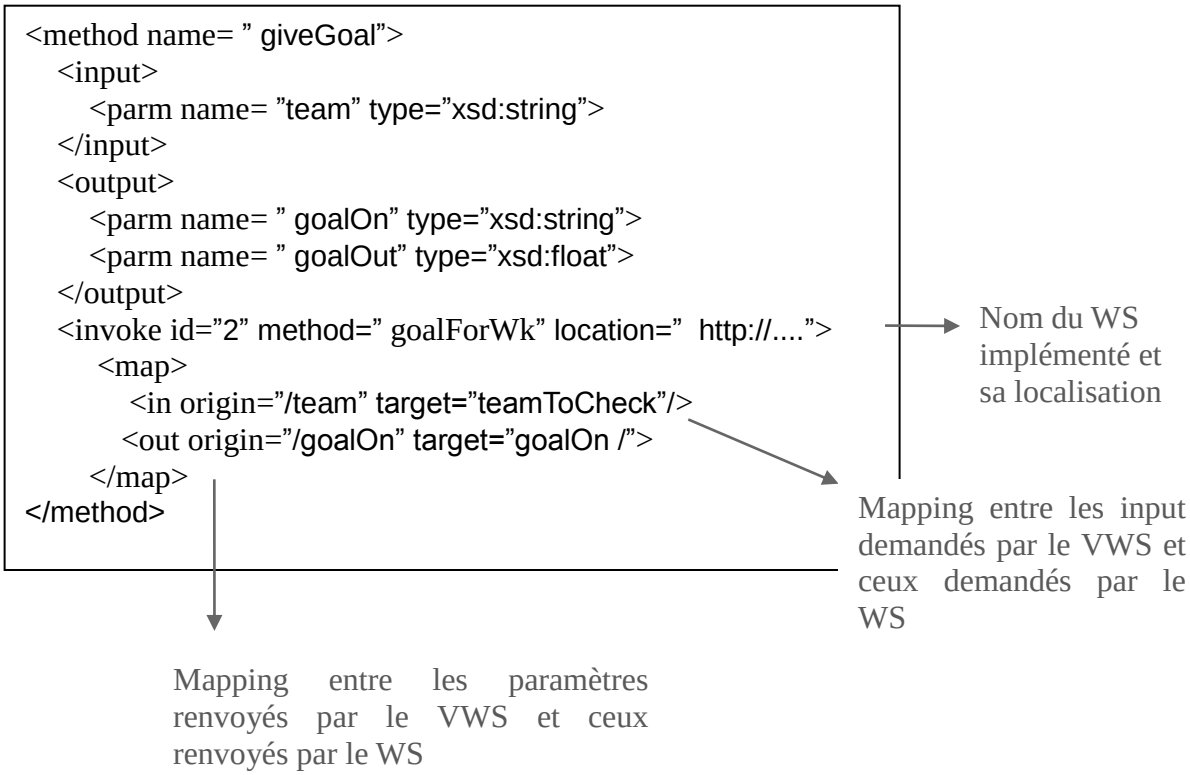


Fig. 12 : Eléments mapping entre un VWS et un WS

Si le web service implémenté correspond parfaitement au VWS, le nœud invoke suffit (le nom de la méthode et la localisation du web service).

Dans le cas contraire, un nœud map doit également être présent.

Comme indiqué dans la figure ci-dessus, ce nœud permet de faire le mapping entre les éléments du VWS et du WS. Tous les paramètres n’ont pas besoin d’être mappés. Il peut très bien y avoir plus ou moins d’éléments renvoyés.

Il y aura autant de nœuds invoke que de fournisseurs offrant un web service implémentant le service proposé par le VWS. Un même fournisseur pouvant également proposer plusieurs instances de son web service, rejoignant par là la réplication géographiquement dispersée que nous avons évoquée lors du chapitre 3.3.3.

Nous voyons donc qu’il y a tout un travail de la part du fournisseur à faire en amont de l’utilisation du VWS : il doit d’abord trouver le Virtual Web Service qui correspond au web service qu’il propose. Ensuite il faut qu’il comprenne le document VWSDL, qu’il l’adapte pour référencer son web service et qu’il y inscrive le mapping éventuel à faire pour que son web service soit effectivement utilisable.

La décision d’appeler tel service plutôt qu’un autre se fait au moment de l’appel au VWS. Pour rappel, cela se fait de manière transparente pour le client. Pour lui, le VWS est un WS classique

qu'il a appelé comme tel en se basant sur sa description WSDL.

C'est au système de virtual web service que revient la tâche de sélectionner le web service le plus approprié au moment de l'appel. Le plus approprié sous entend que sera sélectionné un web service qui est disponible au moment de l'appel. Pour cette sélection du service le plus approprié, 3 éléments sont nécessaires :

- Une mesure de la qualité de chacun des web services du VWS.
- Un historique de la qualité offerte lors des appels précédents.
- Un module en charge de hiérarchiser les web services selon ces critères de qualité.

Chaque fois qu'un web service faisant partie du Virtual Web Service est appelé, des données statistiques de sa qualité de service sont récoltées et conservées. Elles serviront de base de mesure de la qualité lors des appels suivants. Pour chaque VWS, une méthode de classement doit être décidée. Il s'agit d'expression du type :

$$(0.5 * F) + (0.2 * C) + (0.1 * S)$$

Où F représente la fiabilité, C le coût et S le niveau de sécurité.

Des méthodes de conversion sont définies pour rendre ces différents types de mesures compatibles.

Le principe de substitution ici n'est pas de se dire : j'utilise habituellement un certain web service. Lors de l'invocation I, celui-ci n'est pas disponible, j'en recherche donc un autre répondant à mes besoins. Il est plutôt : j'utilise habituellement un certain Virtual Web Service. Le web service réellement appelé derrière m'est caché. Si l'un des web services composant le VWS n'est pas disponible au moment de l'invocation, un autre sera appelé.

Comme on le devine, pour que ce principe fonctionne correctement il faut que suffisamment de fournisseurs de services adhèrent au principe. Si un seul fournisseur de service s'enregistre dans un Virtual Web Service, il n'y a pas de substitution possible.

Un des gros avantages de l'utilisation de la virtualisation dans le cadre des web services est le découplage avec le client. Sans virtualisation, le moindre changement au niveau des paramètres dans le web service implique une correction du côté du client. Ici, seul le mapping entre le web service et le Virtual Web Service doit être adapté. La charge de travail est donc uniquement du côté du fournisseur qui doit corriger le document VWSDL pour que les changements soient effectifs et ce, de manière totalement transparente pour l'utilisateur du VWS.

Le fait que la qualité de service soit prise en compte pour trouver le meilleur substitut est également à l'avantage de cette méthode, de même que le fait que le système de classification utilisé puisse être paramétré en fonction du Virtual Web Service. Comme nous l'avons déjà évoqué, les critères de qualité à prendre en compte et leurs poids dans l'évaluation ne sont pas les mêmes selon les services

rendus, il est dès lors intéressant de pouvoir adapter l'évaluation de ces critères.

Substitution externe via communauté de service

Selon (Maamar et al. 07), une communauté de services est un moyen de fournir une description d'une fonctionnalité désirée sans explicitement faire référence à un web service spécifique.

Une communauté rassemble un ensemble de services répondant à un même ensemble de besoins fonctionnels (Fauvet & al. 07) (Ernst et al. 06). Les services d'une même communauté diffèrent entre eux sur les aspects non-fonctionnels. Ils offrent donc la même fonctionnalité et diffèrent par la qualité de services qu'ils proposent (Maamar & al. 11).

Le principe d'une communauté de services est le suivant. Un besoin fonctionnel est identifié, une communauté est alors créée. Tous les services qui seront rassemblés dans cette communauté rendront dès lors le même service avec des qualités de services variables, dépendants d'un service à l'autre.

Une interface doit être rédigée pour décrire les services que rend la communauté et auxquels s'engagent ses membres. Il est possible et même probable que certains membres ne puissent pas offrir tel quel le service proposé par la communauté. Un adaptateur doit dès lors être écrit pour faire le mapping entre le service tel que présenté par la communauté et le service effectivement implémenté.

Chaque communauté définit un modèle de qualité. Ce modèle décrit les critères de qualité pris en compte au sein de la communauté.

Un maître est désigné au sein de la communauté. Deux approches sont possibles : soit le maître est élu au sein des web services faisant partis de la communauté, soit un service maître est implémenté pour remplir cette fonction.

Les rôles des maîtres de communauté sont multiples (Fauvet & al. 07), (Limam & al. 11):

- D'une part ce sont eux qui sont chargés de recruter de nouveaux membres dans la communauté. Ils consultent pour cela régulièrement les registres UDDI pour repérer les services ajoutés récemment qui pourraient convenir à la communauté.
- Ce sont eux qui sont aussi chargés d'éjecter régulièrement des services de la communauté : ceux offrant une qualité de service insuffisante et n'étant dès lors pas régulièrement choisis pour répondre aux demandes arrivant à la communauté.
- Ils sont chargés de retenir les membres dans la communauté.
- Ils sont également chargés de répartir les demandes arrivant à la communauté entre les

services disponibles.

- Si la communauté ne contient pas de membres, elle est supprimée par le maître de communauté.
- Et enfin des relations avec d'autres communautés peuvent être mises en place : Peer Relationship, ce sont des relations entre communautés dont les domaines sont similaires. Specialization Relationship, ce sont des relations entre deux communautés dont l'une est une spécialisation de l'autre.

Voici à quoi ressemble la communauté après que le maître ait été désigné :

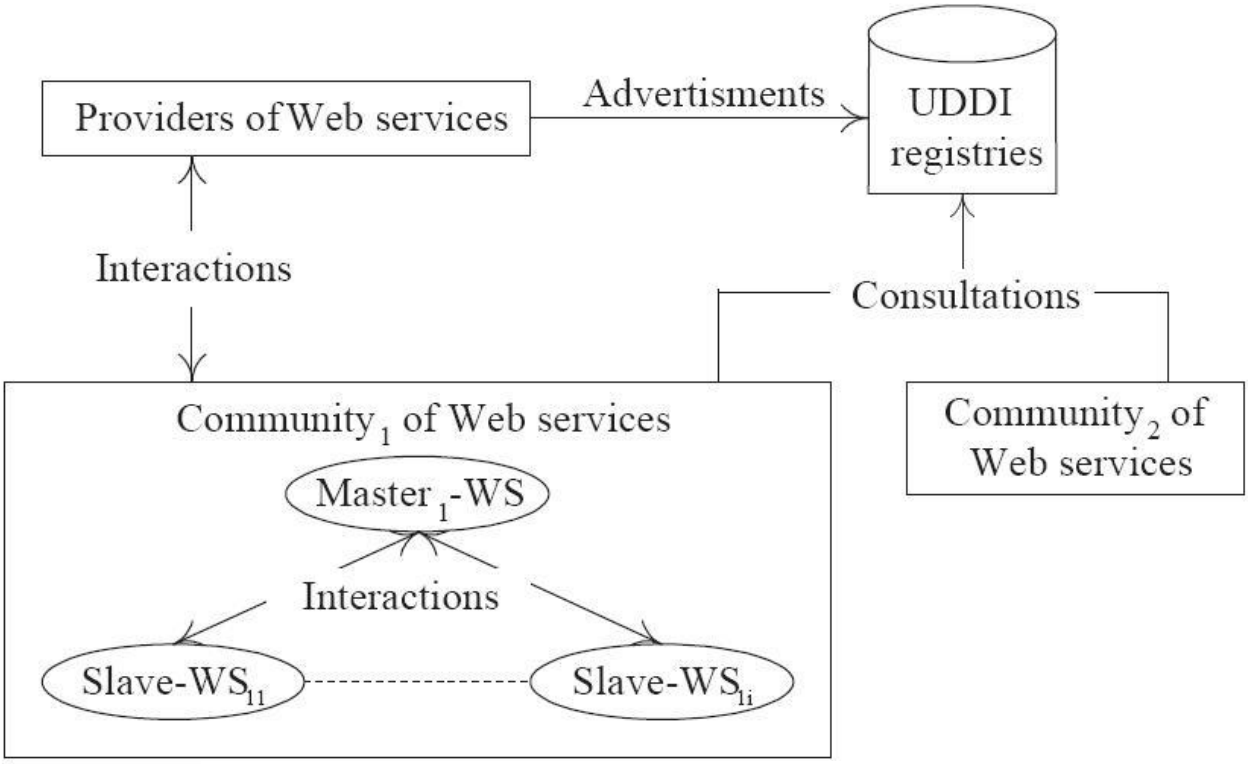


Fig. 13 : Eléments d'une communauté de service (Maamar & al. 07)

Nous avons vu qu'un des rôles du maître de la communauté est d'attirer de nouveaux membres au sein de celle-ci. Mais un web service peut aussi tenter de rejoindre la communauté. Il doit pour cela convaincre le maître qu'il peut apporter quelque chose à la communauté (Bentahar & al. 07), par exemple, une meilleure qualité de service que les autres membres.

Lorsqu'un client veut profiter des services offerts par une communauté, il fait une requête à cette communauté en spécifiant les critères de qualité qu'il souhaite voir rencontrer. Ces critères de qualité doivent être un sous-ensemble de ceux proposés dans le modèle de qualité de la communauté. Sont alors retournés au client la ou les adresses des services correspondant.

De leur coté, les fournisseurs de service peuvent obtenir des informations sur la communauté et ils peuvent également ajouter leurs services à cette communauté.
Nous avons vu que la communauté permet au client de définir quels sont les critères de qualité qu'il souhaite. La communauté de son côté doit être capable de déterminer quel service est le meilleur selon les critères de qualité souhaités.
Elle peut se baser pour cela soit sur une valeur fournie par le fournisseur, soit sur la constitution de trace soit sur l'appel d'une méthode spécifique exposée par le service, les deux dernières étant évaluées au moment de l'appel et du fait plus fiables. Car, d'une part, le reflet de la qualité au moment le plus proche de l'appel et, d'autre part, le fournisseur peut être tenté d'exagérer la qualité de service réellement offerte pour que son service soit plus souvent sélectionné.
Selon (Maamar & al 07), voyons à présent comment s'opère la sélection du web service qui répondra à la demande du client au sein de la communauté.
Quand un client a besoin de la fonctionnalité offerte par une communauté, le maître de la communauté est contacté.
Le maître va alors envoyer un appel à tous les membres qui implémentent la fonctionnalité exposée par la communauté. Cet appel précise les critères de qualité exigés par le client.
Les web services qui sont intéressés à prendre en charge la demande du client et qui rencontrent les critères de qualité exigés répondent au maître.
Celui-ci choisit alors le meilleur parmi les répondants et tous sont notifiés de la décision : celui choisi et les autres. Le web service ayant été choisi se tient alors prêt à répondre à la demande.

La communauté de service permet de préparer une substitution statique telle que nous l'entendons dans le cadre de ce mémoire. En effet, le client de la fonctionnalité offerte par la communauté doit réitérer son appel à celle-ci si le web service renseigné n'est, entre temps, plus disponible.
Son grand avantage est de laisser au consommateur le choix des critères de qualité de services qu'il souhaite voir pris en compte.

Ceci termine notre tour des méthodes de substitution externe. Nous pouvons à présent faire la synthèse des méthodes de substitution interne et externe et en faire la discussion dans le dernier chapitre.

4. Synthèse

4.1 Tableau comparatif des méthodes de substitution

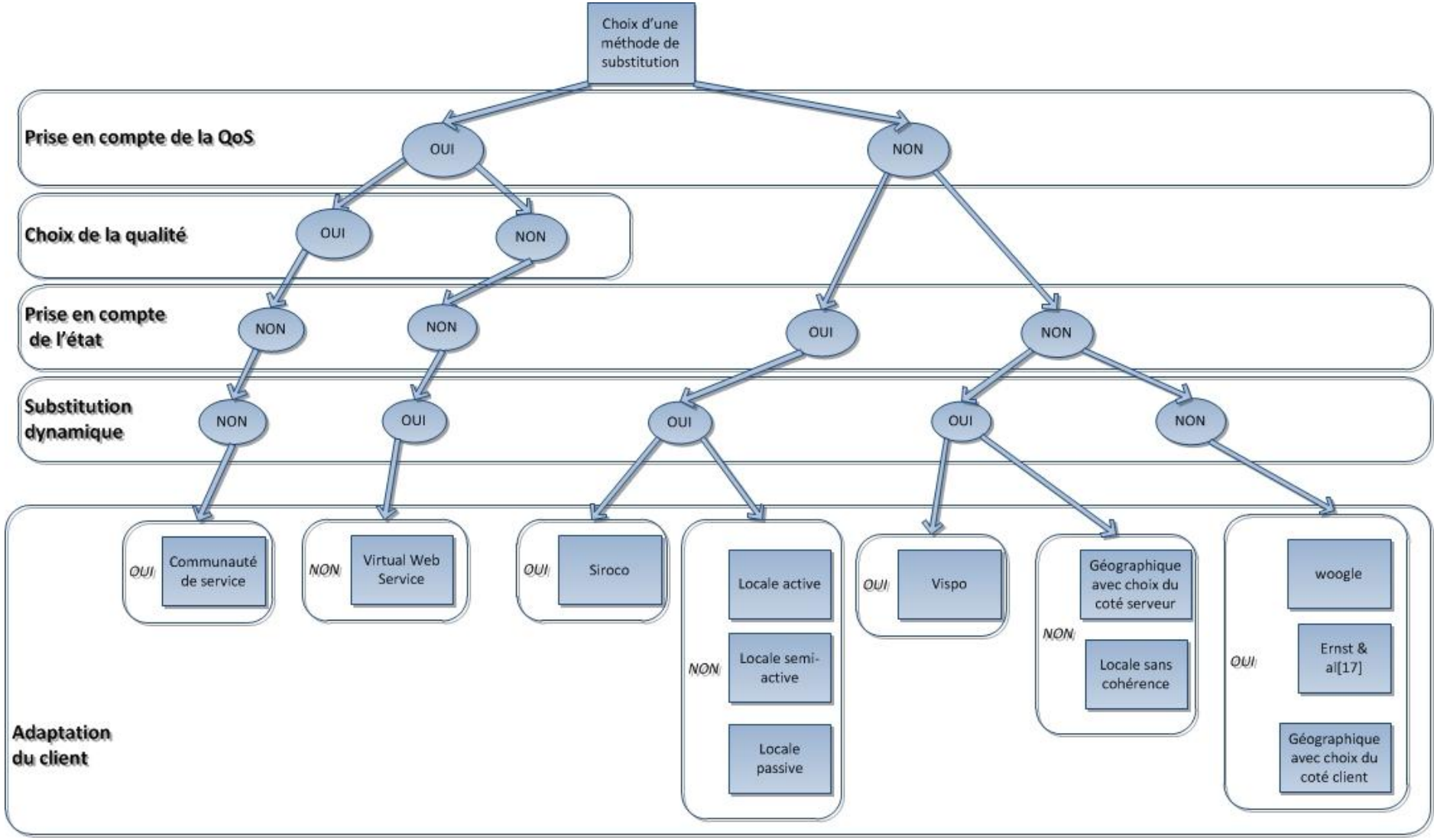
Voici un tableau comparatif des méthodes de substitution interne et externe que nous avons vues tout au long de ce mémoire.

Les éléments repris dans celui-ci seront commentés dans notre discussion.

Nom de la méthode	Nécessité d'une connaissance spécifique de la part du consommateur au sujet de la méthode	Substitution Dynamique	Prise en compte de la QoS lors de la substitution	Le consommateur a le choix de la qualité qu'il souhaite	Prise en compte de l'état du service lors de la substitution
Substitution interne					
Locale sans coherence	Non	Oui	Non mais amélioration du temps de réponse	Sans objet	Non
Locale active	Non	Oui	Non	Sans objet	Oui
Locale semi-active	Non	Oui	Non	Sans objet	Oui
Locale passive	Non	Oui	Non	Sans objet	Oui
Géographique avec choix du côté client	Oui	Non	Non mais amélioration du temps de réponse	Sans objet	Non
Géographique avec choix du côté serveur	Non	Oui	Non	Sans objet	Non
Substitution externe					
(Ernst & al. 06)	Oui	Non	Non	Sans objet	Non
Woogle	Oui	Non	Non	Sans objet	Non
Vispo	Oui	Oui	Non	Sans objet	Non
SIROCO	Oui	Oui	Non	Sans objet	Oui
Communauté de service	Oui	Non	Oui	Oui	Non
Virtual Web Service	Non	Oui	Oui	Non	Non

4.2 Diagramme de choix d'une méthode de substitution

Ce diagramme reprend les critères présentés au point 4.1 mais sous la forme d'un arbre de décision : en parcourant cet arbre l'utilisateur peut, selon les critères qui ont de l'importance pour lui, arriver à la méthode qui correspond à ses besoins.



Comme on le constate en regardant le diagramme, toutes les combinaisons ne sont pas représentées avec les méthodes que nous vous présentons dans ce mémoire.
Aucune méthode ne propose par exemple une prise en compte de la qualité, en laissant le choix de cette qualité, avec prise en charge de l'état du service, en étant un type de substitution dynamique, tout en ne demandant pas d'adaptation du côté du client, ce qui ne veut pas dire qu'il est impossible de trouver dans la littérature une telle méthode. .
Une des combinaisons n'est cependant intrinsèquement pas réalisable : il n'est bien sûr pas possible d'avoir le choix de la qualité de service si la méthode ne propose pas de prise en compte de la qualité.

Afin de bien comprendre le graphique et la discussion qui va suivre dans le chapitre suivant, rappelons ce que nous entendons, dans le cadre de la substitution de service, par les notions d'état et de dynamique.
Dans la grande majorité des cas, un appel à un web service se fait sans que la notion d'état intervienne. C'est-à-dire qu'un programme réalise son appel à un web service et que le résultat de cet appel n'est pas fonction d'un appel antérieur ou d'un appel à un autre web service.
Dans certains cas, par contre, la notion d'état est importante. Ce serait par exemple le cas avec un service de logging (Rao & al. 05). Après l'appel à ce service, l'état ne serait plus not_logged_in mais logged_in et cet appel serait dès lors une pré-condition pour faire appel à d'autres services.
Une méthode qui propose une substitution dynamique, telle que nous l'entendons dans le cadre de ce mémoire, doit se comprendre comme étant une méthode qui choisit le substitut effectivement appelé au moment de l'invocation du web service par le client, sans que celui-ci ait à agir directement sur ce choix.
Maintenant que sont reprécisées ces notions, nous pouvons entamer notre discussion.

5. Discussion

Nous l'avons déjà bien évoqué, l'intérêt de la substitution est d'assurer la disponibilité d'un service ou de profiter d'un service offrant de meilleures qualités de service.

Quand vient la décision de mettre en place une substitution de service, le consommateur peut être intéressé soit uniquement d'avoir son service toujours disponible soit également de prendre également en compte la qualité du service.

Si seule la disponibilité importe, il peut choisir de profiter d'un web service dont le fournisseur a mis en place une stratégie de substitution interne. Trois problèmes peuvent cependant se poser : il n'est pas toujours possible de savoir qu'un fournisseur a mis en place une telle stratégie pour son service, il n'est pas non plus toujours possible de trouver un service avec la fonctionnalité recherchée pour lequel la substitution interne a été mise en place et comme mettre en place une telle stratégie a un coût, il faut parfois être prêt à devoir payer plus cher l'utilisation de ce service. Les avantages générés par cette méthode sont heureusement multiples.

Tout d'abord, le consommateur n'a pas à mettre en place des techniques particulières pour profiter de la substitution, sauf dans le cas de la réplication géographique avec choix du réplica du côté du client. Le consommateur réalise son appel, le fournisseur du service se charge du reste. C'est-à-dire, selon les méthodes, de transmettre cette requête à un ou plusieurs serveurs qui pourront répondre à la demande.

Le consommateur de service peut même espérer, dans le cas de la réplication locale sans gestion de la cohérence et de la réplication géographique avec choix du réplica du côté client, une certaine amélioration du temps de réponse, et donc de la qualité, grâce à la réplication. Pour le premier, du fait qu'est sélectionné le serveur le moins engorgé pour répondre et pour le deuxième, du fait que le client puisse sélectionner le serveur le plus proche pour répondre à ses requêtes.

Si l'état du service doit être pris en compte, nous vous avons proposé trois méthodes : active, semi-active et passive. Aucune de ces méthodes n'est idéale mais chacune a ses avantages. Un certain gain de temps de réponse est à espérer de l'approche active et semi-active puisque la réponse peut être renvoyée dès qu'un des serveurs a traité la demande. L'inconvénient est qu'ils sont beaucoup plus sollicités du fait qu'ils reçoivent toutes les requêtes. Le gain de temps tiré de l'avantage précité peut donc être nul si les serveurs reçoivent beaucoup de demandes.

L'approche passive sollicite beaucoup moins les serveurs mais demande la mise en place d'une gestion des mises à jour des serveurs backup.

Nous n'avons pas trouvé lors de nos recherches de stratégie de substitution interne qui permettrait le choix de la qualité de service. Il est cependant tout à fait envisageable qu'un fournisseur de services propose, pour un même service, différents niveaux de qualité offerte et laisse à l'utilisateur la

possibilité de préciser la qualité qu'il souhaite. Il pourrait alors faire payer l'utilisation de son service d'une somme plus ou moins élevée selon la qualité exigée.
Les stratégies que le fournisseur peut mettre en place pour assurer la disponibilité de son service sont donc multiples et il devrait toujours trouver une solution qui lui convienne. Mais il peut très bien aussi ne pas vouloir la mettre en place, pour éviter les coûts supplémentaires que cela engendre ou il peut juger que ce n'est pas nécessaire pour le type de service qu'il offre.
Le consommateur de son côté peut ne pas être d'accord de payer pour le surcoût engendré par la mise en place de la substitution interne. Il peut, au contraire, être partisan de la substitution interne mais ne pas trouver de service qui l'offre pour la fonctionnalité dont il a besoin.

Il peut alors se tourner vers la substitution externe. Celle-ci peut, en effet, lui être utile pour s'assurer la disponibilité d'un service. Il est, par contre, plus compliqué de trouver le service qui assurera cette disponibilité : trouver deux services offrant la même fonctionnalité tout en ne demandant pas d'adaptation du client pour utiliser l'un à la place de l'autre n'est souvent pas possible. Nous avons cependant pu constater que les propositions ne manquaient pas pour faire face à ce défit, et que chacune apportait ses avantages.

Si seuls les aspects fonctionnels sont à prendre en compte, quatre méthodes peuvent faire l'affaire : (Ernst & al. 06), woogle, SIROCO et VISPO.
Les méthodes proposées par (Ernst & al. 06) et woogle permettent de mettre en place une substitution statique sans que le consommateur doive chercher lui-même manuellement dans les registres UDDI des web services compatibles. Les deux méthodes faisant le travail de manière automatique, une vérification de la part du consommateur des résultats renvoyés peut se révéler bénéfique. Des web services syntaxiquement proches seront en effet renvoyés comme pouvant être substitut l'un de l'autre par les deux méthodes. Mais s'ils diffèrent dans leurs sémantiques, ces méthodes ne le détecteront pas.

Afin d'avoir la disponibilité de son service assurée de manière dynamique, les architectures VISPO et SIROCO sont à prendre en compte. L'architecture VISPO a comme avantage de proposer qu'une personne experte soit responsable de classifier les web services selon leurs sémantiques communes. On peut dès lors raisonnablement penser que les résultats des substitutions sont très fiables. L'action de cette personne peut également être vue comme un désavantage, dans le sens où cette personne devant sans doute être rémunérée d'une façon ou d'une autre, le coût d'utilisation de web services obtenus grâce à cette méthode s'en ressentira.
La méthode SIROCO a l'énorme avantage, par rapport aux autres substitutions externes que nous

vous proposons, d'être la seule à prendre en compte l'état du service à substituer. Elle constitue dès lors le seul choix possible si l'état du service est une notion importante. Par contre, sa méthode de détection des substituts possibles d'un web service se rapproche de la méthode woogle. Les mêmes critiques sont donc émises à son propos : deux méthodes qui diffèrent par leur sémantique mais pas par leur syntaxe seront proposées de façon erronée comme substitut l'une de l'autre.

Si, en plus des aspects fonctionnels, la qualité de service doit aussi être prise en compte, il faut se tourner vers d'autres propositions.
Pour pouvoir substituer un service par un autre en prenant en compte sa qualité de service, nous pouvons nous tourner vers les services web virtuel ou vers la communauté de service.
Un des avantages des services web virtuel est que le client n'a rien à faire pour profiter de la substitution. Il se contente d'appeler le virtual web service comme un service classique et celui-ci se charge d'appeler le service web disponible en son sein qui offre la meilleure qualité de service, avec le désavantage que le client n'a pas le choix des critères de qualité pris en compte.
L'autre avantage de cette méthode est que la substitution est dynamique : le client fait son appel et si un des web services du VWS n'est pas disponible, il n'entre pas en ligne de compte dans le choix du web service effectivement appelé. Comme pour toute méthode proposant une substitution dynamique, il faut dès lors s'attendre à un délai de réponse plus important que pour un appel statique à un web service.

La communauté de service propose aussi une gestion de la qualité de ses membres. Elle impose même une charte de qualité à tous les services qui la rejoignent. Ils doivent au moins être capables de respecter cette charte pour en faire partie.
L'avantage qu'a la communauté de service est de permettre à ses utilisateurs de lui renseigner un sous-ensemble de critères de qualité de la charte à prendre en compte lors des requêtes que ceux-ci vont lui transmettre. La communauté renvoie alors ses membres qui correspondent le mieux à ces critères. Le choix final du web service à appeler est alors laissé à l'utilisateur. Un autre avantage de la communauté est d'être la seule à inclure dans son principe de fonctionnement un mécanisme pour pouvoir attirer de nouveaux membres, ce qui augmente les chances d'avoir des substituts disponibles et qu'ils soient de qualité suffisante pour le consommateur puisque tous les membres doivent respecter la charte.

Et le gagnant est …

Il n'est, bien entendu, pas question de gagnant ou de perdant ici. Comme nous venons de le voir, chaque méthode a ses spécificités, ses avantages et inconvénients. Elles ne sont pas toutes utilisables dans le même cadre.
Certaines ont besoin de connaitre du succès pour que le principe de substitution puisse fonctionner : c'est le cas de la communauté de service, de sirocco, de vispo et du service web virtuel. En effet, toutes ces méthodes se basent soit sur des registres propres soit sur les registres UDDI mais alors uniquement sur un sous-ensemble de web services connu d'elles.
Dès lors, si elles n'ont pas beaucoup de membres, il n'y aura potentiellement pas deux web services rendant la même fonctionnalité, quelles que soit leurs qualités de service pour le coup.

Grâce à cette discussion et au tableau et diagramme qui l'ont précédée, nous voyons que l'essentiel est que l'utilisateur définisse les critères qui sont importants pour lui et que de là il choisisse la méthode qui lui corresponde.

6. Conclusion

En commençant par décrire ce qu'est un web service, quelle est son utilité et comment sont décrites ses caractéristiques fonctionnelles, nous avons progressivement pu nous intéresser à des notions plus avancées, comme la qualité de service, qui permet de décrire et juger un web service sur un autre aspect que ses caractéristiques fonctionnelles.

En poussant ainsi notre réflexion sur les web services, nous avons pu trouver des moyens de pallier le fait qu'un web service utilisé n'offre plus la qualité de service que le consommateur attend de lui, qu'il est tout-à-coup dans l'impossibilité de répondre aux requêtes qui lui sont adressées ou tout simplement que le consommateur veuille le remplacer par un web service offrant une meilleure qualité de service. Des moyens qui prennent en compte les aspects fonctionnels des web services, d'autres qui prennent en plus en compte la qualité de service.

Ces moyens, ce mémoire vous les a présentés. Ils ont comme dénominateur commun : la substitution. Tout au long de ce mémoire et spécialement dans la synthèse et la discussion finale, nous avons pu nous pencher sur les avantages et inconvénients de chaque méthode, et nous rendre compte que si la substitution interne permet le remplacement par un substitut qui est le même que le service à substituer, elle n'est pas toujours disponible et elle entraine un surcoût. Pour la substitution externe, il est plus difficile de trouver un substitut qui convienne à nos besoins mais elle a l'avantage de pouvoir être mise en place pour substituer n'importe quel service, même si cela suppose de mettre en place des stratégies parfois complexes.

7. Bibliographie

Bentahar J., Maamar Z., Benslimane D. and Thiran P., "An Argumentation Framework for Communities of Web Services", *IEEE Intelligent Systems*, v.22 n.6, pages 75-83, 2007.

Booth D., Haas H., McCabe F., Newcomer E., Champion M., Ferris C. and Orchard D., "Web Services Architecture", http://www.w3.org/TR/ws-arch/, (last revised 11/02/2004) (Date of access 30/06/2011).

Bravo M. and Alvadaro M., "Similarity measures for substituting Web Services", *International Journal of Web Services Research*, 7, pages 1-29, 2010.

Conti M., Gregori E. and Lapenna W., "Quality of service in Internet Web Service : Issues and Solutions ", *Proceedings of the 8th HP-OVUA Plenary Workshop*, 2001.

De Antonellis V., Melchiori M., Pernici B. and Plebani P., "A methodology for e-Service substitutability in a virtual district environment", *Proceedings of the 15th international conference on Advanced information systems engineering*, 2003a.

De Antonellis V., Melchiori M. and Plebani P., "An approach to Web Service compatibility in cooperative processes", *Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, pages 95-100, 2003b.

Dong X., Halevy A., Maldhavan J., Nemes E. and Zhang J., "Similarity Search for Web Services", *Proceedings of the Thirtieth international conference on Very large data bases*, pages 372-383, 2004

Ernst M. D., Lencevicius R. and Perkins J. H., "Detection of web service substitutability and composability", *WS-MaTe 2006: International Workshop on Web Services -- Modeling and Testing*, pages 123-135, 2006.

Fauvet M.-C., Duarte H., Taher Y. and Benslimane D., "Sélection dynamique de services Web – une approche à base de communautés", *Acte du XXVème congrès INFORSID*, 2007.

Fernández Vilas J., Pazos Arias J. and Fernández Vilas A., "High Availability with Clusters of Web Services", *APWeb Vol. 3007Springer*, pages 644-653, 2004b.

Fernández Vilas J., Pazos Arias J. and Fernández Vilas A., "An architecture for building Web Services with Quality-of-Service features", *5th International Conference on Web-Age Information Management (WAIM)*, 2004a.

Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P., Berners-Lee T., "Hypertext Transfer Protocol -- HTTP/1.1", http://www.ietf.org/rfc/rfc2616.txt (last revised 06/99) (Date of access 29/08/2011).

Fredj M., Georgantas N., Issarny V. and Zarras A., "Dynamic Service Substitution in Service-Oriented Architectures", *IEEE Congress on Services - Part I*, pages 101-104, 2008.

Huang A. F.M., Lan C. and Stephen J.H. Yang, "An optimal Qos-based Web service selection scheme", *Information Sciences*, number 179, pages 3309–3322, 2009.

Lee K., Jeon J., Lee W., Jeong S.-H. and Park S.-W., "QoS for Web Service : Requirements and Possible Approaches", *W3C Working Group Note*, 2003

Le petit Larousse grand format, édition 2004.

Limam H. and Akaichi J., "Managing and querying web services communities: a survey", *International Journal of Database Management Systems (IJDMS)*, Vol.3, No.1, 2011.

Maamar Z., Benslimane D., Thiran P. and Sattanathan S., "Web services communities – concepts & operations –", 2007.

Maamar Z., Thiran P. and Bentahar J., **"**Web Services Communities:

From Intra-Community Coopetition to Inter-Community Competition", *E-Business Application for Product Development and Competitive Growth: Emerging Technologies*, pp. 333-343, IGI Global, 2011.

Mani A. and Nagarajan A., "Understanding quality of service for Web services", http://www.opensourcetutorials.com/tutorials/Server-Side-Coding/Administration/quality-web-services/page2.html, (last revised 26/12/2006) (Date of access 29/10/2011).

Osrael J., Froihofer L. and Goescha K. M., "What Service Replication Can Learn from Object Replication Middleware", *ACM*, Volume: 184, pages 18-23, 2006.

Osrael J., Froihofer L., Weghofer M. and Goeschka K. M., "Axis2-based Replication Middelware for Web Services", *7ème IEEE International Conference on*

Web Services, 2007.

Parker D., "Understanding the FTP protocol", http://www.windowsnetworking.com/articles-tutorials/network-protocols/Understanding-FTP-Protocol.html, (Last revised 08/09/2005) (Date of access 29/08/2013).

Rao J. and Su X., "Semantic Web Services and Web Process Composition",

Lecture Notes in Computer Science, Volume 3387, pp 43-54, 2005

Shade K., Awodele O., Akinde Ronke O. and Samuel O., "Quality of Service (Qos) Issues in Web Services", *IJCSNS International Journal of Computer Science and Network Security*, VOL.12 No.1, 2012

Taher Y., Benslimane D., Fauvet M.-C. and Maamar Z., "Towards an Approach for Web services Substitution*", International Journal of Web Information Systems*, Vol. 5, pp.32 – 55, 2009

Tere G.M., Jadhav B.T. and Mudholkar R.R., "Dynamic invocation of web services", *Advances in Computational Research*, Volume 4, pages 78-82, 2012

Thirumanan M., Dhavachelvan P., Arbarna S., Arangannayan G., "Architecture for Evaluating Web Service QoS Parameters using Agents", *International Journal of computer Applications*, Volume 10-N.4, 2010.

"About W3C", http://www.w3.org/Consortium/, (Last revised 2012) (Date of access 25/08/2013).

"UDDI executive white paper", http://uddi.org/pubs/UDDI_Executive_White_Paper.pdf, 2001

"XML Tutorial", http://www.w3schools.com/xml/ (Date of access 25/08/2013).

"Introduction to Web Services", www.w3schools.com/webservices/ws_intro.asp (Date of access 25/10/2011).

"WSDL Documents", http://www.w3schools.com/wsdl/wsdl_documents.asp, (Date of access 25/08/2013).