Réutilisation de services web composites par la métaphore du système immunitaire

Rosanna Bova, Salima Hassas, Salima Benbernou

LIRIS (CNRS UMR 5205),
Bâtiment Nautibus,
Université Claude Bernard Lyon I,
43 boulevard du 11 novembre 1918,
69621 Villeurbanne

{rosanna.bova, salima.hassas, salima.benbernou}@liris.cnrs.fr

RÉSUMÉ. *Récemment, plusieurs solutions ont été proposées pour composer des services web. Cependant, peu de solutions connues abordent la question de la réutilisation et de la spécialisation d'une composition de service donnée, c.-à-d., comment des applications peuvent être réalisées par réutilisation, restriction, ou extension de services simples ou composites existants. Dans cet article, nous introduisons la notion de services web composites abstraits, pouvant être à la fois spécialisés et instanciés à des compositions concrètes particulières, et réutilisés dans la construction de compositions plus complexes ou étendues. Nous proposons une approche inspirée des systèmes immunitaires, dans laquelle nous combinons des informations de structure et des informations d'usage, afin de faire émerger des services web composites stables par maturation d'affinité.*

ABSTRACT. *Recently, several web services composition solutions have been proposed. However, few existing solutions address the issue of service composition reuse and specialization, i.e, how applications can be built upon existing simple or composite web services by reuse, restriction, or extension. In this paper, we introduce the concept of abstract composite web service, that can be specialized to particular concrete compositions, and that can be reused in the construction of larger or extended compositions. We propose an approach inspired by immune systems which combines structural and usage information in order to find and reify stable web services composites by an affinity maturation process.*

MOTS-CLÉS : *services web, composition de service, réutilisation de services web composite, système immunitaire.*

KEYWORDS: *web services, service composition, composite web services reuse, immune system.*

Atelier Systèmes d'Information et Services Web, INFORSID 2006

1. Introduction

Un service web peut être défini comme un système logiciel conçu pour supporter les interactions entre des machines situées dans un réseau, basé sur des protocoles web standard et sur le codage en XML des paramètres et des valeurs de retour.

Ces dernières années, la recherche dans le domaine des services web a été très active. Une grande partie de cette recherche a été consacrée à la composition de services web. L'idée originale de « composition de service » n'est pas vraiment nouvelle dans la conception de logiciel. Elle peut être comparée au concept de bibliothèque exécutable dans beaucoup de langages de programmation. En appliquant cette idée au génie logiciel, il est possible de développer de nouvelles applications en utilisant des composants logiciels préexistants. Ce procédé est désigné par « composition » ou « réutilisation » de composants. Dans la communauté des services web, des efforts importants sont maintenant consacrés à ce problème, qui regroupe la découverte de services web, leur composition et leur exécution.

Dans cet article, nous nous sommes intéressés à étudier comment des services web peuvent se composer pour fournir des services web composites, en réutilisant des structures préexistantes : *les composites abstraits*. Ces composites abstraits seront spécialisables en composites concrets particuliers, ou réutilisables dans la construction de composites abstraits plus complexes ou étendus. Le premier type de réutilisation peut être assimilé à une approche descendante, alors que le deuxième est plutôt une approche ascendante, où des composites plus élémentaires sont utilisés comme modules dans des composites plus complexes. Cet article vise à fournir un cadre de travail et un système pour réaliser ces deux types de composition. En utilisant notre cadre de travail, une partie du processus de composition sera réalisée par des ingénieurs ou des administrateurs, le reste étant automatisé par notre système composeur de services web.

2. Définition du problème

Notre but est de définir un système composeur de services web où la composition se base sur la réutilisation et la spécialisation de web services composites. La réutilisation s'appuie sur l'utilisation de méta informations de structure et d'usage exprimant respectivement les liens de spécialisation entre composites *abstraits* ou entre composites *abstraits* et *concrets* et les liens de composition, et le taux de réussite d'utilisation d'un composite dans les compositions antérieures. Ce taux de réussite est une sorte de feedback sur la pertinence de l'usage d'un composite dans un contexte donné. Il est calculé selon un mécanisme inspiré de la notion d'affinité utilisé dans les systèmes immunitaires. Nous détaillerons ces mécanismes dans la suite de cet article.

Nous considérons ici qu'un service composite *concret* est un service web qui appelle d'autres services web (composites ou élémentaires) pendant son exécution. Ces services web référencés sont structurés comme dans un script ou programme qui utilise des opérateurs d'alternatives (conditions), de boucles, de parallélisme et de

séquence. Par exemple, un script BPEL définit un tel service web composite concret. De l'extérieur, un service web composite concret est un service web normal exécutable, accessible en utilisant les mêmes protocoles.

Dans le contexte de la réutilisation de services web composites, nous voyons un service web composite comme une boite grise (*grey box)* plutôt qu'une boite noire (*black box)*, dans le sens où une partie de sa structure interne est accessible à notre système. Pour pouvoir définir la similitude et la compatibilité des fonctionnalités entre plusieurs services web, nous ferons référence à un langage de description sémantique comme OWL-S et une ontologie comme OWL. Définir et utiliser un tel langage et une ontologie associée est en soi un problème difficile, qui sort du cadre de cet article. Cette ontologie sera associée à un répertoire de services web, ou à un moteur de recherche, capable de sélectionner un ensemble de services web (composites ou non) compatibles avec une description sémantique donnée. Ainsi, pour simplifier, nous considérerons que la requête d'un utilisateur est une description sémantique du service web souhaité, sans considérer de contraintes ou de préférences additionnelles de l'utilisateur. Notre travail se focalise sur la façon dont des services web composites abstraits sont spécialisés ou réutilisés, plutôt que sur leur correspondance avec une description sémantique donnée.

Nous supposerons également que notre système ne crée pas de nouvelles compositions, à partir de zéro, mais que des composites concrets et abstraits existent déjà, créés par des concepteurs ou des architectes du système. Notre but est alors de réutiliser ces composites existants, et éventuellement de les adapter au contexte en substituant un composant (service web référencé) par un autre qui lui soit compatible.

Pour cela, nous introduisons le concept de *service web composite abstrait*, basé sur les motifs de haut niveau de services web de (Melloul *et al.*, 2004). Dans un composite abstrait, une ou plusieurs références de service web concret sont remplacées par une ou plusieurs descriptions sémantiques, c.-à-d. par une description de haut niveau des fonctionnalités accomplies par ce composé (similaire à une requête), en utilisant OWL. À chaque description ne peut correspondre qu'un composé. Lors d'une instanciation du composite abstrait, la structure de ce composite est conservée, mais les descriptions sémantiques des composants doivent être résolues.

Les motivations derrière ce concept sont les suivantes : d'une part, cela permet la réutilisation par spécialisation, en adaptant des services composites à d'autres contextes, tout en relâchant certaines contraintes spécifiques ; d'autre part, les composites abstraits nous serviront de lien entre les informations de structure et d'usage. Cependant, comme indiqué par (Melloul *et al.*, 2004), la définition de composites abstraits doit être traitée avec précaution : un composite trop abstrait peut être inutilisable, car il contiendra trop peu d'informations concrètes, et sera très difficile à spécialiser et à adapter à un contexte donné. A l'inverse, il sera difficile de réutiliser un composite abstrait très spécifique dans des situations différentes. Notons que ce compromis entre ces deux extrêmes, permettant une réutilisation suffisante et utile à la fois, se retrouve dans toute problématique de réutilisation de composants.

Dans un premier temps, une autre hypothèse de ce travail est que les composites concrets contiendront des références fixes de service web. Les composites concrets ou abstraits ne pourront par exemple pas inclure dans leur exécution un processus de recherche de services web.

En conclusion, notre problème est de trouver des services web composites appropriés et utilisables, si possible des composites *stables*, par rapport à une requête d'utilisateur donnée. Ce composite pourra être un service composite concret existant, ou la spécialisation automatique d'un service web composite abstrait.

3. Scénario : agence de voyage virtuelle

Nous utiliserons l'exemple de l'agence de voyage comme scénario. Considérons les services web suivants :

– Des services web de réservation de billet de voyage : *avion1* (qui couvre les vols depuis et vers Londres), *avion2* (qui couvre les vols depuis et vers Paris), *avion3* (qui couvre les vols depuis et vers Milan et Trento), et *Eurostar* ;

– Des services web de réservation d'hôtel : *hôtel Paradise*, *hôtel Coconut* ;

– Des services web de location de voiture : *voiture1*, *voiture2* ;

– Un service web composite concret très spécifique appelé *VoyageDeParisA-Londres* (cf. figure 1), qui combine l'exécution d'un service web de réservation du train *eurostar* depuis Paris, suivi de la réservation de l'*hôtel paradise* à Londres, et de la location d'une voiture avec *voiture2*. Le rectangle arrondi représente ce composite, avec ses entrées (*période*) et sorties (*billet de train*, *réservation d'hôtel* et *location de voiture*) ;

– Un service web composite abstrait appelé *VoyageALondres* (cf. figure 2), représentant la généralisation du service composite concret précédent, où le service *eurostar* est remplacé par la description sémantique *BilletVoyage*, avec un paramètre d'entrée, *destination*, fixé à la valeur Londres, et où le service *hôtel paradise* est remplacé par la description sémantique *ReservationHotel* ;

– Un autre composite abstrait, appelé *VoyageDirect*, abstraction de *VoyageA-Londres* et, par transitivité, de *VoyageDeParisALondres*. Ici toutes les références de services web impliqués sont remplacées par les descriptions sémantiques abstraites *BilletVoyage*, *ReservationHotel* et *ReservationVoiture*, avec pour entrées *origine*, *destination*, *période*, et les mêmes sorties que les deux composites précédents.

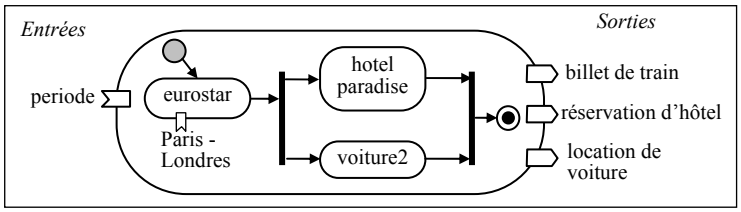

Figure 1. *Diagramme d'activité UML représentant VoyageDeParisALondres*

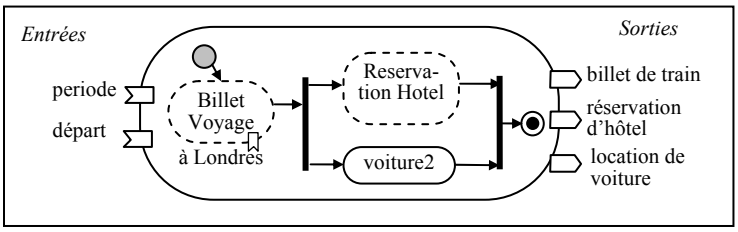

Figure 2. *Diagramme d'activité UML de VoyageALondres*

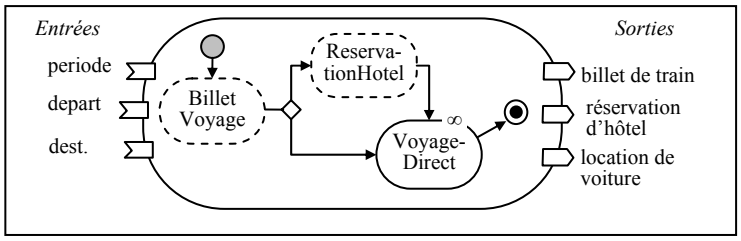

Figure 3. *Diagramme d'activité UML de VoyageIndirect*

– Enfin, un dernier composite abstrait (cf. figure 3), appelé *VoyageIndirect*, incluant une référence au composite abstrait *VoyageDirect*, comme indiqué par le symbole ∞ sur la figure 3. Une réservation d'hôtel peut être nécessaire si le voyage s'étale sur une nuit dans une ville intermédiaire, représenté ici par l'alternative.

4. Approche inspirée des systèmes immunitaires

4.1. *Vue simplifiée d'un système immunitaire*

Un des rôles d'un système immunitaire (De Castro *et al.*, 1999) est de protéger le corps contre les attaques des pathogènes, appelés aussi antigènes. Les mécanismes employés par le système immunitaire (SI) à cette fin sont :

– la reconnaissance de forme de l'antigène (*Pattern Recognition)*, qui est effectuée par des récepteurs, les *paratopes*, sur la surface des anticorps libérés par les cellules immunitaires (cellules B). La partie sur l'antigène en contact avec un paratope s'appelle *epitope*. La liaison d'un antigène aux différents anticorps nécessite que leurs structures aient des formes complémentaires. La force de cette liaison dépend de l'affinité entre antigène et anticorps ;

– la réponse immunitaire (*Immune Response*), constituée par une réponse immédiate à l'attaque de l'antigène d'une part, puis une réponse adaptative d'autre part, spécifique à l'antigène et liée au processus de maturation d'affinité (voir ci-dessous) ;

– la sélection clonale (*Clonal Selection*). Quand les anticorps se lient à l'antigène, la cellule B s'active et commence à proliférer. De nouvelles cellules B sont produites, copies conformes de la cellule B parente, et spécifiques à l'antigène envahissant ;

– la maturation d'affinités (*Affinity Maturation*) qui garantit que le système immunitaire devient de plus en plus efficace face aux attaques des antigènes. Après la réponse immédiate, une certaine quantité de cellules B demeure et joue le rôle de mémoire immunologique. Cela permet au système immunitaire de lancer une attaque plus rapide et plus forte quand l'antigène attaque une deuxième fois. Cette deuxième réponse plus rapide est attribuée aux cellules de mémoire restantes dans le système immunitaire, qui ont de plus amélioré leur affinité par rapport à cet antigène par hyper mutation somatique ou à travers le processus de sélection clonale.

4.2. *Métaphore du Système Immunitaire pour la réutilisation de composites*

Bien que les systèmes immunitaires aient beaucoup inspiré notre approche, notre ambition n'est pas de définir une correspondance exacte entre les concepts, les processus et les mécanismes de notre modèle et ceux du système immunitaire. Nous utiliserons cette métaphore autant que possible, particulièrement quand elle aide la compréhension, toutefois certaines spécificités de notre solution n'ont pas de contreparties dans les systèmes immunitaires, et vice versa.

Système Immunitaire	Système composeur de services web
Antigène	Requête de l'utilisateur
Reconnaissance de forme	*Matching* des descriptions sémantiques et services web existants + choix de l'utilisateur
Affinité	Compatibilité sémantique + affinité relative et globale
Maturation d'affinité	Election et réification des services web composites stables
Cellules mémoire	Services web composites stables
Cellules B	Services web composites concrets ou virtuels

Tableau 1. *Tableau récapitulant les correspondances entre les principaux mécanismes du système immunitaire et ceux de notre modèle. Certains des termes utilisés ici sont détaillés dans la section 5.*

5. Modèle et principe de fonctionnement

Pour atteindre nos objectifs, nous proposons de combiner *l'information de structure et l'information d'usage*, afin de favoriser, de faire émerger, et éventuellement de publier les services web composites *stables*.

Ces composites stables représenteront des solutions potentiellement réutilisables dans de nouvelles compositions correspondant à une catégorie des requêtes, de la même manière que les cellules mémoire dans un système immunitaire réagissent à une certaine catégorie d'antigènes.

Nous définissons *l'information de structure* à partir des relations liant entre eux les services web référencés (représentant les composants des composites) et organisés dans un processus bien défini. Le détail de cette information de structure dépend du langage de description du *workflow* ou de processus employé pour décrire les services web composites. Enfin, l'information de structure comporte aussi les relations de généralisation/spécialisation entre composites.

Nous définissons *l'information d'usage* par une métrique dans notre modèle : l'*affinité relative*, associée à une relation entre chaque composant d'un service web composite, et les diverses abstractions de ce dernier.

Nous supposons l'existence d'un répertoire local référençant un ensemble de services web composites ou simples. En plus des entrées, des sorties et des opérations produites par un web service, nous ajoutons dans ce répertoire les méta informations de structure – les relations de généralisation et de dépendance de composition – et d'usage – les relations d'affinité relative.

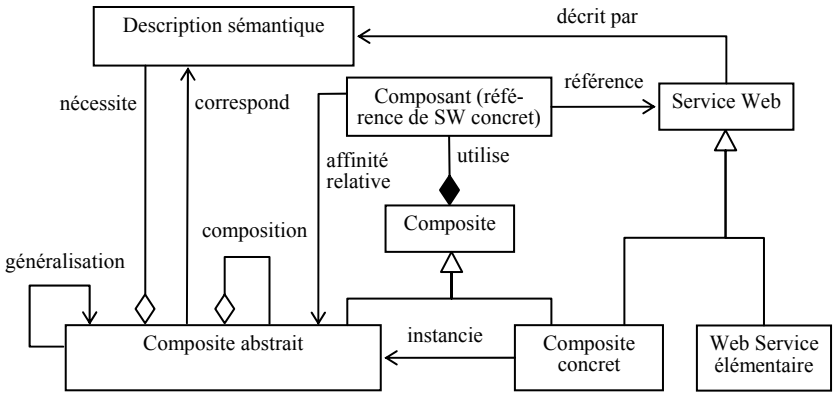

Figure 4. *Diagramme de classe UML des meta informations de structure et d'usage*

La figure 4 présente un diagramme de classe UML décrivant notre modèle. A un service web est associée une description sémantique. Un composite représente une structure organisant des composants (références de services web composés), qui peuvent être composites ou élémentaires. Un composite n'est lui-même un service web que s'il est concret, c'est à dire exécutable. Un composite abstrait comprend à la fois des composants concrets, des descriptions sémantiques (à instancier), et peut référencer d'autres composites abstraits par composition. Il correspond également à une description sémantique.

Les relations de structure que nous exploiterons sont la relation de généralisation et d'instanciation (cette dernière est le maillon terminal de la chaîne de généralisation), la relation de composition entre composites abstraits (il s'agit d'une dépendance de composition pour être précis), et celle réifiée par l'entité Composant. La relation d'affinité relative, valuée, représente l'information d'usage.

5.1. *Fonctionnement du système*

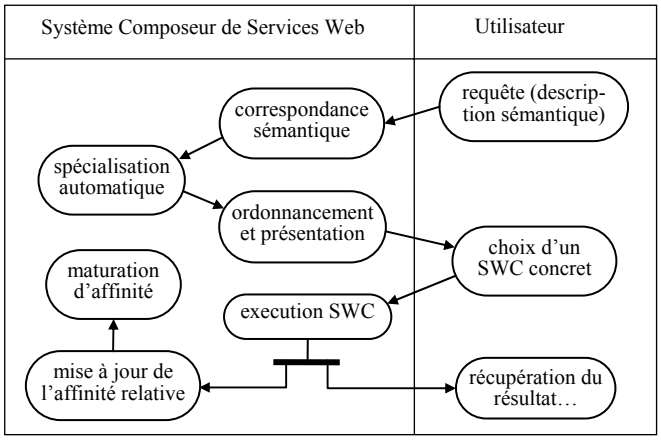

Figure 5. *Processus de composition et d'exécution du composite*

La figure 5 représente le processus de fonctionnement de notre système. Comme indiqué dans la section 2, nous considérons qu'une requête utilisateur est équivalente à une description sémantique, et qu'il existe un mécanisme de correspondance sémantique, entre elle et les candidats compatibles, parmi les services web composites et les services élémentaires présents dans le répertoire. Le système composeur réalise d'abord la spécialisation automatique, qui consiste à générer (par spécialisation), à partir des composites abstraites compatibles, et en utilisant l'affinité relative comme guide, de nouveaux composites concrets potentiels. Cette tâche ne crée pas des composites à partir de zéro, mais explore de nouvelles possibilités d'instanciation.

Le système ordonne ensuite les propositions de composition générées, ainsi que d'autres composites candidats concrets obtenus lors de la mise en correspondance sémantique, en utilisant une valeur globale d'affinité pour chaque composite, et présente le résultat à l'utilisateur. Ce dernier choisit une proposition. Le composite choisi est alors exécuté par le système, et les valeurs d'affinité relatives sont mises à jour. L'utilisateur choisira éventuellement un nouveau composite concret exhibant une haute affinité globale, et s'il répond aux conditions d'un composite stable (c.-à-d. de cellule de mémoire, défini section 5.3), le processus de maturation d'affinité lui associera une description sémantique, de sorte qu'il devienne sélectionnable par le mécanisme de correspondance sémantique. Ce composite sera alors utilisable dans des spécialisations automatiques futures.

5.2. *Affinité relative*

Définition : Affinité Relative. Une valeur d'affinité relative est toujours associée à une relation entre un service web concret (composite ou pas), et un composite abstrait du composite qui l'appelle. Cette valeur est mise à jour chaque fois que ce

service web concret est exécuté en tant que composant d'une des spécialisations de ce composite abstrait (ce dernier représente le contexte d'utilisation du composant). La fonction d'affinité relative se défini comme suit :

$$aff_r(c,a) = \frac{freq_{succ}}{freq_{usage}} \quad [1]$$

$freq_{succ}$ est une fonction qui mesure le nombre de fois que le service web concret (c) a été employé avec succès, et $freq_{usage}$ est le nombre total de tentatives d'utilisation de ce service web concret, dans le contexte du composite abstrait.

5.3. *Maturation d'affinité*

Pendant l'étape de spécialisation automatique, un nouveau composite concret virtuel peut être créé en spécialisant les composites abstraits compatibles avec la requête de l'utilisateur. Un composite virtuel n'est par défaut pas considéré comme stable par le système : il n'a pas d'existence propre en dehors de la session courante de l'utilisateur, et n'est ni identifié ni associé à une description sémantique dans le répertoire de services web du système composeur. La seule trace de son existence est représentée par la valeur d'affinité globale calculée par le système, et utilisée par la suite pour ordonner les propositions avant de les présenter à l'utilisateur. Cette valeur d'affinité globale dépend seulement des valeurs d'affinité relatives des différents composants impliqués dans le composite virtuel.

Définition 1 : Affinité Globale. La valeur de l'affinité globale est une moyenne pondérée des différentes valeurs d'affinité relatives, par rapport aux composites abstraits ancêtres successifs du composite concret. Sa valeur est indiquée par :

$$aff_g(C) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\alpha_{ij} aff_r(c_i, a_i^j)}{n} \quad [2]$$

C est le composite concret considéré, vu comme un ensemble de n composants, références de services web concrets, $(c_1 \ldots c_i \ldots c_n)$; m représente le nombre de composites abstraits ancêtres de C ; a_i^j sont les descriptions sémantiques référencées dans le $j^{ème}$ ancêtre abstrait de C, instanciées par les composants c_i ; α_{ij} représentent des pondérations associées aux services web concrets c_i (la somme de ces valeurs se rapportant au même service web concret descendent est égale à 1); enfin $aff_r(c_i, a_{ij})$ est l'affinité relative de c_i par rapport à a_i^j.

Définition 2 : Composite Virtuel. Un composite virtuel est un composite créé par le système composeur comme résultat de l'étape de spécialisation automatique, en réponse à une requête utilisateur. Ce composite est *temporaire* à la session, et n'a pas d'existence propre dans le répertoire de service web.

Définition 3 : Composite Stable. Un composite stable est un composite initialement virtuel qui a été employé avec succès au moins une fois, et qui présente une valeur d'affinité globale excédant un seuil prédéfini (paramètre du système), que nous désignerons par seuil de maturation d'affinité.

5.4. *Application au scénario de l'agence de voyage*

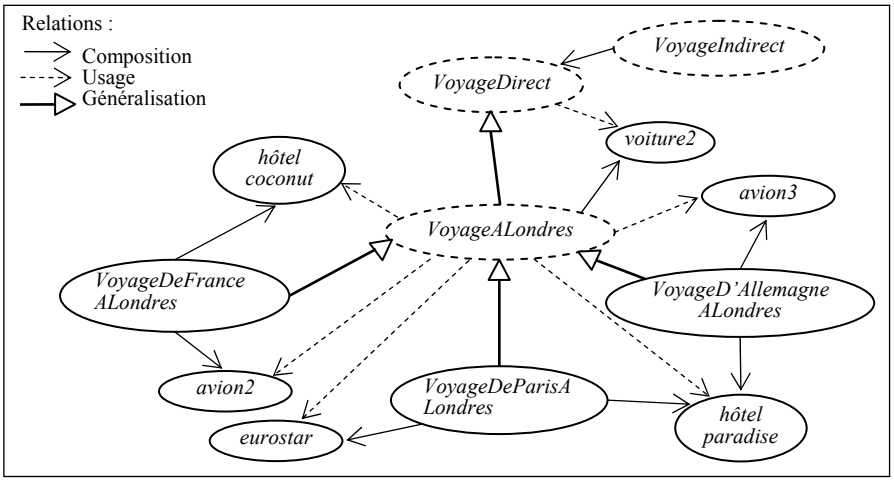

Figure 6 *Exemple de l'agence de voyage*

Pour illustrer notre approche, nous supposons qu'un utilisateur souhaite aller à Londres depuis la France, dans le contexte du scénario présenté dans la section 3, et augmenté dans la figure 6. Nous supposons que la requête de l'utilisateur est compatible avec *VoyageDeFranceALondres*. Cependant, nous supposons également que *VoyageDeParisALondres* et *VoyageD'AllemagneALondres* ont été utilisés largement avec succès, de sorte que la valeur qui mesure *l'affinité relative* d'*hôtel paradise* en ce qui concerne le composite abstrait *VoyageALondres* soit plus haute que celle d'*hôtel coconut*. Bien que le contexte soit légèrement différent, *hôtel paradise* est encore applicable à la description sémantique *ReservationHotel* inclus dans *VoyageALondres*. Ainsi, le système composeur proposera à l'utilisateur un composite virtuel avec *hôtel paradise* au lieu de *hôtel coconut*, avec une valeur de *préférence* (*affinité globale*) plus élevée. Maintenant, nous supposons que l'utilisateur choisit ce service web composite virtuel, et que l'exécution réussit. Si l'affinité globale est plus grande que le seuil de maturation d'affinité fixé à l'avance par le constructeur du système, un nouveau composite concret stable est créé et référencé dans le répertoire, ce dernier pouvant ensuite être utilisé directement.

Nous pourrons constater que l'émergence du composite stable n'est pas due à la seule utilisation du composite virtuel créé par le système, mais aussi par l'utilisation croisée de composites proches partiellement compatibles avec la requête.

6. Etat de l'art

Cette approche s'inspire du travail de (Melloul *et al.*, 2004). En particulier, le concept de composite abstrait introduit ici est proche de la notion de motifs de composition définie dans cet article.

Bien que nous n'utilisons pas explicitement de notion de classes, les relations de généralisation/spécialisation suggèrent un modèle d'héritage. Dans la littérature, différents travaux traitent de l'héritage de classe de *workflow*. Par exemple, (Bussler, 1999) présente un tel *framework* et analyse les besoins afin de définir ce type d'héritage, en comparant différentes perspectives et en proposant un langage de définition de classe de *workflow*. (Kappel *et al.*,1995) présentent des spécifications de classe de *workflow*, qui consistent en un ensemble de classes d'objets et un ensemble de règles. Dans (Papazoglou et al.,2002) ,un système appelé TOWE met en application un ensemble de classes de base qui fournissent des mécanismes d'exécution de *workflow*. Des classes de *workflow* peuvent y être développées par héritage à partir de ces classes de base.

Notre approche diffère des travaux sur la composition automatique de services web par le fait qu'elle se focalise sur la réutilisation croisée de composites abstraits existants, soit par instanciation directe en composites concrets particuliers, soit dans la construction de composites plus étendus ou complexes.

D'autre part, la relation de généralisation que nous considérons est définie par la substitution, dans la définition d'un composite, d'un ou plusieurs composants (services web concrets) par des descriptions sémantiques. Il ne s'agit pas ici d'une méta information supplémentaire, indépendante de la structure des composites.

Enfin, notre travail se distingue de celui de (McIlraith et al., 2001) sur la composition sémantique de services web, et de celui de (Petrie et al., 2004) sur la planification de service web, pour lesquels l'objectif est de produire un plan de composition. A l'inverse, nous partons de « plans » de haut niveau existants et nous nous concentrons sur leur possible réutilisation, en exploitant la combinaison des informations de structure et d'usage, représentées par les relations d'affinité et les relations de composition et de généralisation.

7. Conclusions et perspectives

Dans cet article nous avons proposé une approche permettant la réutilisation de services web composites, la spécialisation automatique de services web abstraits, à travers l'instanciation de composants à forte affinité relative, et l'émergence de composites stables, en s'inspirant du système immunitaire. La correspondance de formes entre les *epitopes* de l'antigène et les *paratopes* des anticorps est représentée, dans notre système, par une fonction d'affinité relative qui mesure le degré de succès d'usage d'un service web concret, référencé par un composite concret, par rapport au service web composite abstrait de ce composite, qui représente d'une certaine manière le contexte d'utilisation du composant. Le processus associé de *maturation d'affinité* permet l'apparition de nouveaux composites concrets stables, qui résultent

de spécialisations automatiques guidées par l'information d'usage accumulée. Ces composites concrets stables sont alors identifiés et sémantiquement décrits, tels que n'importe quel service web existant dans notre système, afin de leur conférer une existence propre, et de pouvoir les publier et les exploiter dans d'autres contextes.

La première perspective de ce travail est de définir un prototype afin de valider la faisabilité de cette approche sur des scénarios simples. Il serait aussi intéressant de considérer la spécificité de la requête dans notre évaluation d'affinité globale : si une requête est très spécifique, il est raisonnable de penser que les valeurs d'affinité relative liées aux composites abstraits les plus spécifiques soient plus importantes que les valeurs d'affinité relative liées aux plus abstraites. A l'inverse, une requête très vague s'intéressera moins aux premières, et plus aux dernières.

Enfin, une perspective à plus long terme est d'exploiter la nature distribuée des processus du système immunitaire, et sa tolérance naturelle à l'hétérogénéité. Au lieu d'avoir un système global de composition, cette approche peut s'appliquer à un réseau de domaines de compositions de services web interconnectés, contrôlés par des systèmes composeurs locaux. Ces composeurs locaux pourraient par exemple publier et s'échanger les descriptions sémantiques des nouveaux composites stables, selon une politique de diffusion donnée, reproduisant ainsi la diffusion des cellules mémoire dans notre sang.

Références

Bussler C., « Workflow class inheritance and dynamic workflow class binding », *In Proceeding of the Workshop Software Architectures for Business Process Management at the 11th Conference on Advanced Information System engineering,*, Heidelberg, Germany, 1999.

De Castro L. N., Von Zuben F. J., Artificial Immune Systems: Part I, Basic Theory and Applications, RT DCA Technical Report, vol. 1, n° 98, 1999.

McIlraith S. A., Son T. C., and H. Zeng, « Semantic Web Services », *IEEE Intelligent Systems, Special Issue on the Semantic Web*, vol. 16, n° 2, 2001, p. 46-53.

Melloul L., « A. Fox Reusable Functional Composition Patterns for Web Services », *IEEE International Conference on Web Services,* vol. 498, 2004.

Kappel G., Lang P., Rausch-Schott S., Retschitzegger W., Workflow Management Based on Objects, Rules and Roles, Bulletin of Technical Committee on data Engineering, Vol.18, 1995.

Papazoglou M. P., Yang J., « Design Methodology for Web Services and Business Processes », *Procedings of the 3rd VLDB-TES Workshop*, Hong Kong, August 2002. Also Lecture Notes in Computer Science, Springer, Vol. 2444, 2002.

Petrie C., Genesereth M., Bjornsson H., Chirkova R., Ekstrom M., Gomi H., Hinrichs T., Hoskins R., Kassoff M., Kato D., Kawazoe K., Min J. U., Mohsin W., *Adding AI to Web Services*, Agent Mediated Knowledge Management, LNAI 2926, Springer, 2004, p. 322-338.