Patrons transactionnels dynamiques pour des services composés fiables et flexibles

Imed Abbassi *, Mohamed Graiet **, Éric Cariou ***, Zied Jaoua ****

*ENIT, Université de Tunis El Manar
abbassi_imed@ymail.com
**ISIMM, Université de Monastir, Tunis
mohamed.graiet@imag.fr
***LIUPPA / Université de Pau et des Pays de l'Adour, France
Eric.Cariou@univ-pau.fr
****ISIMM, Université de Monastir, Tunis
jaouazied2002@yahoo.fr

Résumé. Dans cet article, nous proposons une solution pour spécifier une composition dynamique fiable et flexible. Pour y parvenir, nous introduisons un nouveau concept appelé les patrons transactionnels dynamiques. Ce nouveau concept est une convergence entre les patrons workflows dynamiques et les modèles transactionnels avancés. Il peut être considéré comme une coordination dynamique et comme une transaction structurelle. Nous intégrons l'expressivité et la puissance des patrons de workflow dynamique et la fiabilité des modèles transactionnels avancés. Nous utilisons un ensemble de règles pour assurer une composition dynamique fiable et flexible de services Web transactionnels.

1 Introduction

Les services Web peuvent être définis comme des programmes modulaires, indépendants et qui peuvent être découverts, localisés et invoqués à travers Internet en utilisant un ensemble de protocoles standards (Tidwell et al., 2001). Un des concepts les plus intéressants est la possibilité de créer dynamiquement de nouveaux services Web à valeur ajoutée en sélectionnant ceux qui répondent mieux aux exigences des clients puis en les combinant (Mustafa et McCluskey, 2009). L'objectif de ce travail est d'assurer la fiabilité d'exécution des services Web composés dynamiques. La fiabilité d'exécution est un aspect important dans un environnement dynamique et qui n'a pas été profondément examiné.

Les technologies connexes actuelles sont incapables de résoudre ce problème de manière efficace. Ces technologies reposent sur deux approches existantes : les modèles transactionnels avancés (Elmagarmid, 1991) et le système de workflow (W. M. P. van der Aalst et Kiepuszewski., 2000). Les modèles transactionnels avancés permettent d'assurer une exécution correcte d'un ensemble d'opérations encapsulées à l'intérieur d'une unité de traitement appelée transaction. Les systèmes de workflow sont focalisés sur la coordination statique (pas de

configuration dynamique) et l'aspect organisationnel des processus métiers. Ces deux technologies sont incapables d'assurer des reconfigurations dynamiques fiables d'une composition de services Web. Dans (Graiet et al., 2013; Imed et al., 2013), nous avons proposé une approche basée sur la preuve en utilisant la méthode B événementielle pour assurer la fiabilité des services Web composés dynamiques transactionnels. Dans cet article, nous proposons une solution pour assurer des compositions dynamiques fiables et flexibles de services Web. Pour y parvenir, nous introduisons un nouveau paradigme, appelé patron transactionnel dynamique, pour spécifier des compositions dynamiques fiables de services Web. Ce nouveau concept est une convergence entre les patrons de workflows dynamiques (Lam, 2008) et les modèles transactionnels avancés. Notre approche intègre l'expressivité et la puissance des patrons de workflow dynamique et la fiabilité des modèles transactionnels avancés (ATM). Nous utilisons un ensemble de règles et une grammaire pour résoudre le problème de fiabilité d'exécution et la cohérence du flot de contrôle d'une composition dynamique de services Web.

Dans ce qui suit, nous présentons certains travaux connexes (section 2) et l'exemple fil rouge de l'article (section 3). Dans la section 4, nous introduisons le modèle de DTCS (Dynamic Transactional Composite Service). Dans la section 5, nous proposons des patrons de contrôle flot dynamique qui sont par la suite étendus avec des dépendances transactionnelles. Nous montrerons comment nous utilisons les patrons transactionnels dynamiques pour spécifier un DTCS fiable et cohérent.

2 Travaux connexes

Le workflow (Jablonski et Bussler, 1996) est devenu une technologie clé pour l'automatisation des processus métier, offrant un support pour les aspects organisationnels, l'interface de l'utilisateur, la distribution et l'hétérogénéité (Alonso et al., 1996).

Dans (van Der Aalst et al., 2003), les auteurs proposent une collection de patrons de flot de contrôle statique qui peuvent être classés en six groupes : les patrons de flot de contrôle de base, les patrons de branchement multiple et de synchronisation, les patrons structurels, les patrons d'instanciation multiple, les patrons à base d'état et les patrons d'annulation. Contrairement aux (van Der Aalst et al., 2003) et (Russell et al., 2006) qui se sont focalisés principalement sur les patrons de workflow statiques, nous nous concentrons dans notre approche sur les patrons de workflow dynamiques (Lam, 2008). Les patrons de choix exclusif dynamique ne sont pas considérés dans notre approche. En effet, ils sont remplacés par un patron séquence dynamique. Il n'est pas nécessaire de définir d'une manière statique les services Web alternatifs puisque le service qui va remplacer celui qui n'est plus fonctionnel peut être déterminé automatiquement et invoqué à l'exécution. D'autres travaux connexes (Barros et al., 2005; Russell et al., 2005b) présentent des collections des patrons de workflow de ressources et d'interaction de services. En outre, (Russell et al., 2005a) propose une série de patrons workflow de données qui visent à capturer les différentes façons dont les données sont représentées et utilisées.

La fiabilité d'exécution des services Web composites est un aspect stimulant qui n'a pas été profondément examiné. Les technologies connexes actuelles sont incapables d'assurer une exécution fiable. Ces technologies peuvent être classées en deux groupes à savoir : les technologies à base de workflow comme WSBPEl (Andrews et al., 2003) et WS-CDL (Kavantzas et al., 2005) et les technologies qui sont basées sur les transactions comme WS-BusinessActivity (Freund et al., 2005) et WS-TXM (Bunting et al., 2003). Les technologies à base de transac-

tions assurent seulement la fiabilité d'exécution. Cependant, ils sont trop rigides pour supporter des applications basées sur des processus métier comme les services Web composés dans un environnement dynamique. Les systèmes de workflow se concentrent principalement sur la coordination statique et sur les aspects organisationnels et ignorent les problèmes de fiabilité et l'adaptabilité, par exemple l'échec ou la non-disponibilité d'un composant.

Différents travaux existants tentent d'assurer la fiabilité d'exécution des services composites. Le travail présenté dans (Bhiri et al., 2006) propose une solution pour assurer une composition fiable. Les auteurs introduisent un nouveau concept, appelé les patrons transactionnels, pour spécifier facilement des compositions fiables de services Web avec une configuration statique. Ce nouveau concept est une convergence entre les patrons de workflows statiques et les modèles transactionnels avancés. Il peut être considéré comme une coordination qui est définie statiquement et comme une transaction structurelle. La solution ainsi proposée intègre l'expressivité et la puissance des patrons de workflow statiques et la fiabilité des modèles transactionnels avancés (ATM). Elle utilise un ensemble de règles et une grammaire pour résoudre le problème de fiabilité et de cohérence du flot de contrôle, mais dans un environnement stable où les composants disponibles sont stables ou rarement modifiables. La grammaire proposée assure une connexion cohérente d'un ensemble d'instances de patron transactionnel d'un service composite. Ainsi, elle postule que (i) un flux de contrôle cohérent doit commencer par une instance soit d'un patron séquence, soit d'un patron de branchement (ii) une instance d'un patron séquence peut être suivie par n'importe quelle instance d'un patron de branchement (iii) une instance d'un patron AND-split peut être suivie par une instance du AND-join (iv) une instance du patron OR-split peut être suivie par une instance d'un OR-join ou d'un patron XOR-join, et (v) une instance d'un patron XOR-split ne peut être suivie que par un patron XOR-join. En outre, un composant d'un service composite donné peut être lui-même un service composite où son flux de contrôle est conforme (il respecte la grammaire) ; ce qui permet ainsi d'utiliser des patrons d'une manière imbriquée à l'intérieur d'une composition.

Par rapport à (Bhiri et al., 2006), l'avantage principal de notre approche est qu'elle assure non seulement la fiabilité d'exécution, mais aussi la flexibilité d'une composition de services Web dans un environnement (dynamique), où les attentes des clients et les composants disponibles sont variables. Pour cela, nous introduisons un nouveau concept appelé les patrons transactionnels dynamiques pour spécifier des compositions dynamiques fiables de services Web. Ce nouveau concept est une convergence entre les patrons de workflow dynamiques et les modèles transactionnels avancés.

3 Scénario

Nous choisissons d'illustrer notre approche par un scénario (figure 1) dans lequel un client désire organiser un circuit touristique.

Initialement, l'application TCR (Touristic Circuit Reservation) est composée de plusieurs tâches à savoir réservation de vol (T1), réservation de voiture (T2), réservation d'hôtel (T3), paiement en ligne (T4) et enfin envoi des documents (T5). Tout d'abord, le client accomplit la tâche de réservation de vol qui est réalisée par les services Web "Flight Booking" (FB) et "Flight Reservation" (FR) (d'autres services Web peuvent être découverts automatiquement). Ensuite, les tâches de réservation de voiture et d'hôtel peuvent être exécutées en parallèle. Les clients ne sont pas strictement obligés d'effectuer les réservations d'hôtel et de voiture

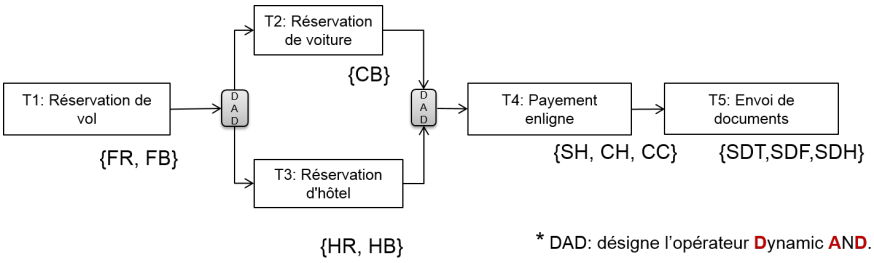

FIG. 1 – *Scénario*

pour leur voyage, donc le service de réservation de véhicule devrait être facultatif. La tâche de réservation de voiture est fournie uniquement par le service Web “Car Booking” (CB). La réservation d’hôtel est exécutée soit par “Hotel Reservation” (HR), soit par le service “Hotel Booking” (HB). Lorsque ces deux tâches sont terminées, alors on demande au client de payer le coût de la réservation. La procédure de paiement est réalisée par plusieurs services Web, selon le mode de paiement, à savoir par carte de crédit “Credit Card” (CC), par chèque “Check” (CH), ou en espèce “caSH” (SH). Finalement, les documents de réservation sont envoyés au client en exécutant l’un des services Web suivants : “Send Document by Fedex” (SDF), “Send Document by DHL” (SDH) et “Send Document by TNT”(SDT), selon le transporteur. Nous remarquons que, pour une simple tâche, plusieurs services peuvent être sélectionnés. Pour gérer cela, nous choisissons une technique multicritère (Alireza Afshari et Yusuff, 2010) basée sur la qualité de service (QoS). Nous notons par S_t^* le service qui sera choisi pour exécuter la tâche t.

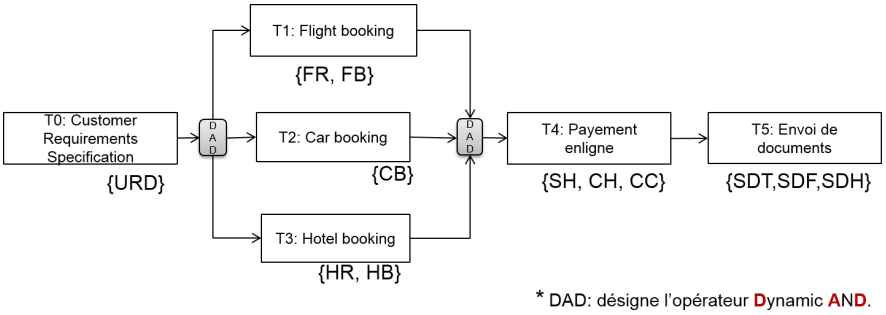

FIG. 2 – *Le modèle de processus du service TCR après la modification*

Des améliorations sont maintenant apportées au service TCR (figure 2). En effet, une activité, “Customer Requirements specification” (CRS) est ajoutée. Elle permet aux clients de spécifier leurs préférences. L’activité CRS peut être réalisée par le service Web “User Requirements Definition” (URD) (d’autres services Web fournissant cette activité peuvent apparaitre ultérieurement). Un client peut également imposer des contraintes sur le coût, le temps d’exécution et la disponibilité des activités proposées. L’ordre d’exécution et le mécanisme de recouvrement sont également modifiés. En effet, l’activité CRS est effectuée juste après avoir

reçu la demande du client. Ensuite, les réservations de vol, d'hôtel et de voiture sont exécutées simultanément et sont synchronisées. Une fois que ces activités se sont terminées avec succès, la procédure d'envoi des documents est effectuée après la terminaison de la tâche de paiement.

Les patrons de workflow standards ne sont pas adaptés pour la modélisation de cet exemple. En effet, les patrons de workflow ne sont pas adaptés à un environnement dynamique où de nouveaux composants (services Web) peuvent être composés à l'exécution et où d'autres peuvent être ajoutés, disparaître, ou être remplacés automatiquement.

4 Comportement transactionnel des services Web composés dynamiques

Dans cette section, nous proposons notre modèle pour assurer une exécution correcte d'un DTCS (Dynamic Transactional Composite Service). Ce dernier intègre l'expressivité, la puissance de modèles de workflow dynamiques et la fiabilité des modèles transactionnels avancés (ATM). L'originalité de ce modèle est la flexibilité offerte non seulement aux concepteurs, mais aussi aux clients pour spécifier leurs besoins.

4.1 Modèle de Service Web transactionnel

Un service Web est un composant logiciel mis à disposition sur Internet, indépendant et autodescriptif, qui peut être automatiquement découvert, localisé et invoqué via des protocoles standards (Snell et al., 2009).

Un service Web transactionnel est un service Web dont le comportement manifeste des propriétés transactionnelles. Les propriétés transactionnelles considérées dans notre approche sont : pivot(p), compensable(c), rejouable(r) (Agrawal et El Abbadi, 1992; Bhiri et al., 2005). Un service Web est dit rejouable s'il se termine toujours avec succès après un nombre fini de réactivations. Un service Web est dit compensable s'il offre des mécanismes de compensation pour annuler sémantiquement son travail (Rollback) tandis qu'un service Web est dit pivot si une fois qu'il se termine avec succès, ses effets ne peuvent pas être compensés. Typiquement, un service Web peut combiner un ensemble de propriétés transactionnelles. L'ensemble de toutes les combinaisons possibles est { $\emptyset$, {r}, {p}, {c}, {r,p}, {r,c}} (Bhiri et al., 2005). En outre, un service web peut disposer d'un ensemble de caractéristiques non fonctionnelles qui est défini par un ensemble d'attributs de QoS (coût, disponibilité, temps de réponse).

4.2 Modèle de service composé transactionnel dynamique

Un service composé dynamique est une conglomération de services Web existants pour offrir un nouveau service à valeur ajoutée (Mustafa et McCluskey, 2009; Jureta et al., 2007). Les services Web qui participent dans le processus de composition sont découverts puis combinés pendant l'exécution pour répondre à un objectif fixé par l'utilisateur. Cette sorte de composition est aussi appelée "composition à base d'objectif". Ainsi, si un composant n'est pas disponible, alors il est possible de le remplacer par un autre qui fournit le même processus métier et qui satisfait les mêmes exigences.

Un service composé transactionnel dynamique (DTCS) est défini comme un service composé dynamique, où ses composants sont des services Web transactionnels (Graiet et al., 2013).

Il exploite non seulement l'avantage des modèles transactionnels avancés, mais aussi la substitution dynamique pour définir les mécanismes de recouvrement en cas d'échec ou de non-disponibilité.

Un DTCS définit un ensemble de préconditions sur l'activation des services Web. Ces préconditions expriment plusieurs sortes de relation de dépendances, à savoir dépendances d'activation, de compensation, d'annulation et d'abandon. Ces relations de dépendances sont définies dynamiquement en exécutant un ensemble de règles de cohérences. Les dépendances d'activation définissent un ordre partiel sur l'activation des services Web. Une dépendance d'activation de S1 vers S2 ne peut exister que si la terminaison de S1 peut déclencher l'activation de S2. Dans la figure 1, les concepteurs définissent une dépendance d'activation de S^*_{T1} vers S^*_{T3} et de S^*_{T1} vers S^*_{T2} sachant que S^*_{T2} et S^*_{T3} s'activent simultanément après la terminaison de S^*_{T1}. Les dépendances transactionnelles spécifient plusieurs types de dépendances à savoir compensation, annulation et abandon. Une dépendance de compensation de S1 vers S2 ne peut exister que si l'échec ou la compensation de S1 peut déclencher la compensation de S2. Une dépendance d'annulation permet de signaler l'échec d'exécution de service vers les autres services qui s'exécutent en parallèle en annulant, si nécessaire, leur exécution. Une dépendance d'annulation de S1 vers S2 ne peut exister que si l'échec de S1 peut déclencher l'annulation de S2. Une dépendance d'abandon permet de propager l'échec (provoquant l'abandon de DTCS) d'un service à tous ses successeurs. Une dépendance de l'abandon de S1 vers S2 ne peut exister que si l'échec, l'annulation ou l'abandon de S1 peut déclencher l'abandon de S2. Comme indiqué dans la figure 1, des dépendances de compensation de S^*_{T2} vers S^*_{T3} et de S^*_{T2} vers S^*_{T1} sont définis, telle que S^*_{T3} et S^*_{T1} seront compensés si S^*_{T2} échoue.

Les relations de dépendance précédemment détaillées doivent respecter quelques règles de cohérence. En effet, les dépendances transactionnelles dépendent des dépendances d'exécution normale. Pour cela, nous considérons les règles suivantes :

1. Une dépendance d'abandon de s1 vers s2 ne peut exister que s'il existe une dépendance d'activation de s1 vers s2.
2. Une dépendance de compensation de s1 vers s2 ne peut être définie que si (i) s2 est compensable, (ii) s1 peut échouer et (iii) il existe une dépendance d'activation de s2 vers s1, ou s1 et s2 s'exécutent en parallèle et sont synchronisés.
3. Une dépendance d'annulation de s1 vers s2 ne peut être définie que si s1 peut échouer et si s1 et s2 s'exécutent en parallèle et synchronisés.

Ces règles sont utilisées dans des travaux précédents pour assurer une composition statique fiable de services Web transactionnels (Bhiri et al., 2005, 2006).

4.3 Flot de contrôle et flot transactionnel

Les dépendances transactionnelles sont totalement définies par les dépendances de compensation, d'abandon et d'annulation entre les composants du DTCS. Les dépendances d'activation et transactionnelles expriment respectivement le flot de contrôle et le flot transactionnel d'un DTCS.

Le flot de contrôle d'un DTCS définit un ordre partiel sur l'activation de services composants. Intuitivement, il est défini par l'ensemble de ses dépendances d'activation. Dans notre approche, nous définissons un flot de contrôle comme un DTCS dans lequel seules des dépendances d'activation sont définies.

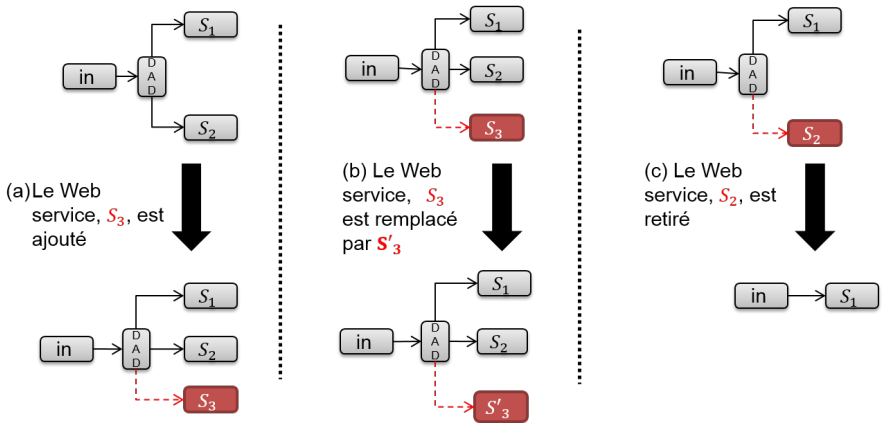

FIG. 3 – *La re-configuration dynamique du patron DAD-split après l'ajout, la suppression ou le remplacement d'un composant*

Le flot transactionnel d'un DTCS définit les mécanismes de recouvrement en cas d'erreur. Intuitivement, il est défini selon les propriétés transactionnelles des composants et de l'ensemble des dépendances transactionnelles. Dans notre approche, nous spécifions le flot transactionnel comme un DTCS dans lequel seules des dépendances transactionnelles sont définies.

5 Patrons de flot de contrôle dynamique

L'utilisation des patrons du workflow dynamiques (Lam, 2008) semble être une idée pertinente pour composer à l'exécution un ensemble de services Web. Contrairement à (van Der Aalst et al., 2003; Russell et al., 2006) qui se concentrent sur les patrons workflows statiques, la structure des patrons de workflow dynamiques est définie lors de l'exécution. Ainsi, les patrons de workflows dynamiques peuvent être considérés comme des patrons de flot de contrôle. Formellement, un patron de flot de contrôle dynamique est défini en tant que fonction qui retourne un flot de contrôle étant donné un ensemble de services.

Dans cette section, nous proposons des patrons de flot de contrôle dynamique pour décrire le flot de contrôle du DTCS. Nous nous focalisons sur les trois patrons suivants : DAD-split (Dynamic AnD-split), DAD-join (Dynamic AnD-join) et DSQ (Dynamic SeQuence) pour expliquer et illustrer notre approche. Toutefois, les concepts présentés ici peuvent être appliqués à d'autres patrons à savoir DOR-split (Dynamic OR-split), DOR-join (Dynamic OR-join) et DNM-Join (Dynamic N out Of M Join).

Le patron DAD-split est défini comme un point dans un procédé du workflow dynamique où un flot de contrôle se divise en plusieurs flots de contrôle en parallèle, permettant ainsi une exécution parallèle d'un ensemble de processus. Il fournit un flot de contrôle dynamique. En effet, un sous-processus simple peut être soit ajouté (figure 3.a), retiré (figure 3.c), ou remplacé par un autre (figure 3.b). La Figure 3.c présente que le patron DAD-split peut se transformer en un patron séquence.

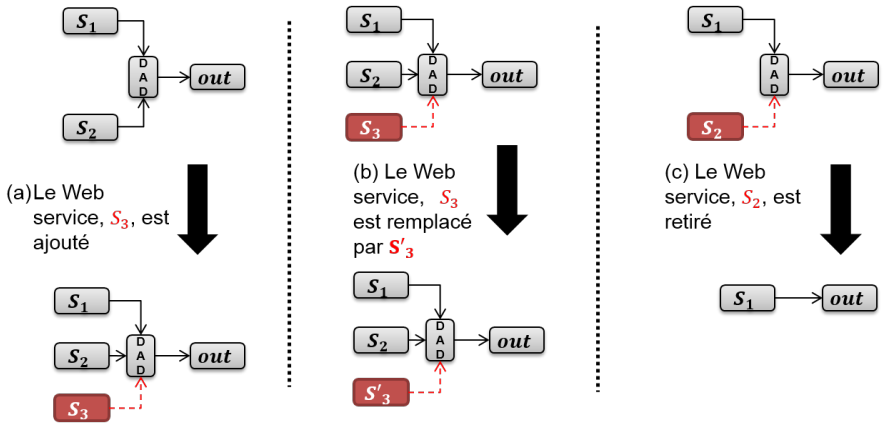

FIG. 4 – *La re-configuration dynamique du patron DAD-join après l'ajout, la suppression ou le remplacement d'un composant*

Le patron DAD-join est défini comme un point dans un procédé du workflow dynamique où plusieurs flots de contrôle en parallèle convergent et se synchronisent en un seul processus d'exécution. Il fournit un flot de contrôle dynamique. En effet, un sous-processus simple peut être soit ajouté (figure 4.a), retiré (figure 4.c), ou remplacé par un autre (figure 4.b). Comme montré dans la Figure 4.c, un patron DAD-join peut se transformer en un patron séquence.

Le patron DSQ est défini comme un point dans un procédé du workflow dynamique où un sous-processus déclenche l'exécution d'un autre. Il fournit un flot de contrôle dynamique. En effet, un sous-processus simple peut-être soit remplacé (figure 5.a) soit ajouté. Dans le deuxième cas, un patron DSQ peut se transformer soit en un patron DAD-split (figure 5.b),

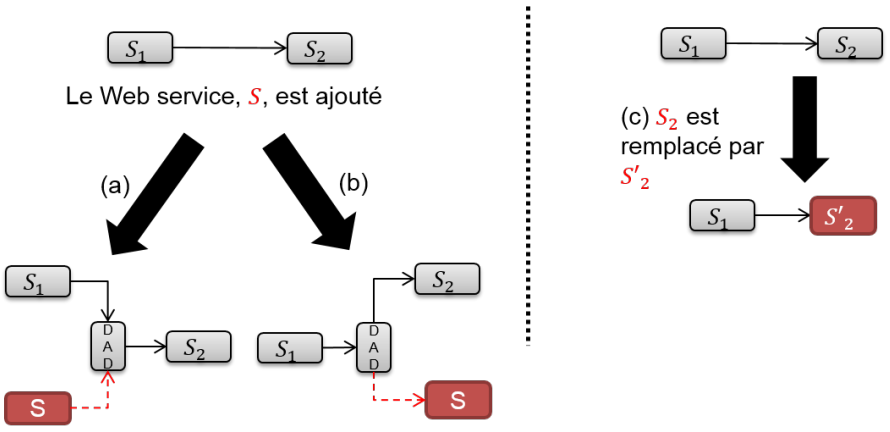

FIG. 5 – *La re-configuration dynamique du patron DSQ après l'ajout, ou le remplacement d'un composant*

soit en un patron DAD-join (figure 5.c).

6 Composition fiable utilisant des patrons transactionnels dynamiques

Dans cette section, nous étendons le patron de flot de contrôle dynamique avec des dépendances transactionnelles. En particulier, nous montrons comment ces patrons transactionnels dynamiques sont utilisés pour modéliser la composition dynamique de services Web.

6.1 Patrons transactionnels dynamiques

Etant donné des propriétés transactionnelles des services Web et des patrons de flot de contrôle dynamiques, nous avons pu déduire de nouveaux patrons, appelés patrons transactionnels dynamiques, qui seront utilisés pour spécifier à la fois le contrôle et les flux transactionnels. Un patron transactionnel dynamique peut être considéré comme une coordination dynamique et comme une transaction structurelle. Ainsi, il définit non seulement le flot de contrôle, mais le flot transactionnel. Il combine la flexibilité de flot de contrôle dynamique et de la fiabilité des modèles transactionnels avancés. Dans ce travail, nous présentons les patrons transactionnels dynamiques suivants DTAD-split (Dynamic Transactional AnD split), DTAD-join (Dynamic Transactional AnD join) et DTSQ (Dynamic Transactional SeQuence).

A partir d'un patron transactionnel dynamique, plusieurs instances peuvent être définies. Une instance est l'application d'un patron transactionnel dynamique à un ensemble de services Web. Les dépendances transactionnelles de l'instance sont un sous-ensemble de l'ensemble des dépendances transactionnelles du patrons transactionnelle dynamique (chaque patron transactionnel dynamique définit un ensemble de dépendances transactionnelles potentielles). Par exemple, les concepteurs peuvent définir une dépendance d'abandon de S^*_{T2} vers S^*_{T4} selon l'opérateur de flot de contrôle DAD-join parce que l'abandon, l'annulation ou l'échec de S^*_{T2} déclenche l'abandon de S^*_{T4}. De même, il est possible de définir une dépendance de compensation de S^*_{T2} vers S^*_{T1} selon l'opérateur du flot de contrôle DAD-split parce que l'échec de S^*_{T2} nécessite la compensation du travail terminé par le service Web S^*_{T1}.

6.2 Composition

L'utilisation des patrons transactionnels dynamiques semble être une idée intéressante pour composer un ensemble de services Web pour obtenir un service composite transactionnel dynamique (DTCS) qui est fiable du point de vue de l'exécution. Dans notre approche, un DTCS est défini comme un ensemble d'instances de patrons transactionnels dynamiques correctement reliés entre eux (partage de certains services Web). Par exemple, la figure 6 montre comment un DTCS qui est généré dynamiquement pour réaliser la réservation de circuit touristique, est défini comme l'union des instances de patrons transactionnels dynamiques suivantes : DTAD-split(URD,FR,CB,HB) ; DTAD-join(FR,CB,HB,SH) ; DTSQ(SH,SDT).

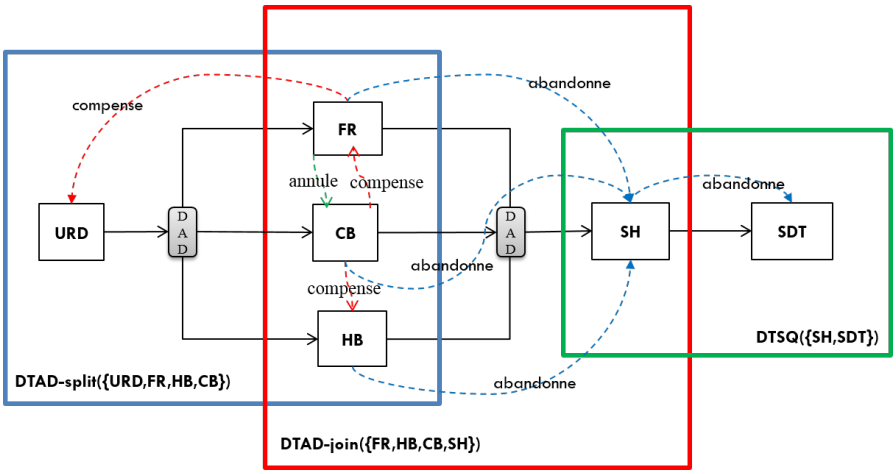

FIG. 6 – *Une connexion cohérente d'un ensemble d'instances de patrons transactionnels dynamiques*

6.3 Cohérence du flot de contrôle et du flot transactionnel

La connexion d'un ensemble d'instances de patrons transactionnels dynamiques peut conduire à une incohérence dans le flot de contrôle et dans le flot transactionnel. Par exemple, le problème de la cohérence du flot de contrôle peut apparaître lorsque les instances sont disjointes, ou incohérentes (avec des sémantiques incompatibles). De même, l'incohérence transactionnelle peut apparaître quand un service Web échoue provoquant l'abandon de tout le DTCS. Par exemple, si nous supposons que SH n'est pas rejouable (il peut échouer) dans le DTCS défini dans la figure 6, cela signifie que HB et CB devraient être compensés afin de s'assurer qu'il n'existe pas un effet restant (la réservation d'hôtel et de voiture sont réalisées) après l'abandon de DTCS. Cela implique qu'il existe deux dépendances de compensation, de SH vers HB et de SH vers CB, et que HB et CB sont compensables.

Pour résoudre ces problèmes, nous définissons une composition dynamique comme valide que si elle assure à la fois la cohérence du flot de contrôle et du flot transactionnel. Afin de ne pas avoir des instances disjointes ou des unions d'instance du patron incohérentes (avec des sémantiques incompatibles), nous proposons une grammaire contextuelle (tableau 1). Nous considérons une union d'instances de patron de flot de contrôle dynamique comme un mot dont les terminaux sont les instances et le symbole "$\cup$". Cette grammaire définit le langage des unions cohérentes. Un flot de contrôle est cohérent si et seulement si l'union des instances correspondantes ou une union équivalente est un mot généré par cette grammaire. Deux unions sont équivalentes si elles sont constituées des mêmes instances de patron.

En utilisant cette grammaire, nous assurons une connexion cohérente des instances de patron de flot de contrôle dynamique d'un DTCS. Ainsi, elle postule que :

- un flot de contrôle doit commencer par une instance d'un des patrons DSQ, DAD-split, ou DOR-split.
- une instance du patron DSQ ou d'un patron de synchronisation est suivie par une instance d'un des patrons DSQ, DAD-split ou DOR-split.

$$
\begin{array}{lcl}
G_i & \longrightarrow & DSQ(G_1, G_2) \cup A \mid \\
 & & DAD - split(G_1, G_{11}, ..., G_{1n}) \cup B \mid \\
 & & DOR - split(G_1, G_{11}, ..., G_{1n}) \cup C \\
A & \longrightarrow & DSQ(G_2, G_3) \cup A \mid \\
 & & DAD - split(G_2, G_{21}, ..., G_{2n}) \cup B \mid \\
 & & DOR - split(G_2, G_{21}, ..., G_{2n}) \cup C \mid \epsilon \\
B & \longrightarrow & DAD - join(G_{11}, ..., G_{1n}, G_3) \cup A \mid \\
 & & DOR - join(G_{11}, ..., G_{1n}, G_3) \cup A \\
 & & DNM - join(G_{11}, ..., G_{1n}, G_3) \cup A \\
C & \longrightarrow & DOR - join(G_{11}, ..., G_{1n}, G_3) \cup A \\
 & & DNM - join(G_{11}, ..., G_{1n}, G_3) \cup A
\end{array}
$$

TAB. 1 – *Grammaire contextuelle définissant le langage des flots de contrôle dynamiques cohérents*

– une instance du patron DAD-split doit être suivie par une instance d'un des patrons DAD-join, DOR-join ou DNM-join.
– une instance du patron DOR-split doit être suivie par une instance d'un patron DOR-join.

En outre, un composant (service Web) d'un DTCS donné peut être lui-même un service composite où son flot de contrôle est cohérent (c'est-à-dire, il respecte la grammaire). Ceci permet d'utiliser des patrons de contrôle de flot dynamiques d'une manière imbriquée à l'intérieur d'une composition.

Afin de garantir la fiabilité d'un DTCS, nous utilisons un ensemble de règles qui sont précédemment introduites (section 4.2). Brièvement, l'algorithme que nous utilisons est comme suit :

1. Etape 1 : après l'échec ou la non-disponibilité d'un composant S1, nous cherchons un nouveau service Web, S2, s'il existe, pour substituer S1.
2. Etape 2 : si la substitution de S1 échoue, alors nous annulons toutes les activités des services qui s'exécutent en parallèle.
3. Etape 3 : ensuite, nous compensons ce qui peut être compensé (seulement les activités réalisées) selon les dépendances de compensation définies.
4. Etape 4 : si le client a annulé la requête, alors nous annulons les services Web en cours d'exécution.
5. Etape 5 : enfin, nous compensons les services Web qui terminent leurs exécutions.

7 Conclusion et perspectives

Dans cet article, nous proposons une approche pour assurer une composition dynamique de services Web fiables. L'idée principale est de combiner l'adéquation des systèmes de workflow

dynamique et la fiabilité des modèles transactionnels avancés. Nous introduisons le concept de patron transactionnel dynamique. Les patrons transactionnels dynamiques s'étendent des patrons de workflow dynamiques avec des dépendances transactionnelles, permettant ainsi de réduire leur manque transactionnel. Nous montrons comment nous les utilisons pour définir des services Web composites dynamiques et comment nous assurons leur fiabilité.

La contribution principale de ce travail est la convergence de l'approche de workflow dynamique et les modèles transactionnels avancés. Outre les intérêts théoriques, notre approche a des avantages pratiques. Quant aux perspectives de ce travail, nous envisageons les prolongements suivants :

- formaliser en B événementiel les patrons transactionnels dynamiques pour mieux exprimer leurs sémantiques ;
- proposer une stratégie efficace pour l'interprétation des règles de cohérence transactionnelle et l'automatisation de la composition de services Web transactionnelle. Cette stratégie est basée sur les heuristiques. Intuitivement, elle permet de réduire l'espace de recherche des solutions possibles tout en garantissant en même temps des solutions optimales. Il est donc nécessaire de définir un critère de comparaison des compositions transactionnelles possibles conformes aux règles de cohérence.
- proposer des re-configurations dynamiques fiables pour faire évoluer les services composites pendant leur exécution tout en préservant leur disponibilité ;
- étudier le problème d'implémentation de l'approche proposée en utilisant les standards et les technologies connexes.

Références

Agrawal, D. et A. El Abbadi (1992). Transaction management in database systems. In *Database Transaction Models for Advanced Applications*, pp. 1–31. Morgan Kaufmann.

Alireza Afshari, M. M. et R. M. Yusuff (2010). Simple additive weighting approach to personnel selection problem. In *International Journal of Innovation, Management and Technology, Vol. 1, No. 5*, pp. 166–173.

Alonso, G., D. Agrawal, et A. El Abbadi (1996). Process synchronization in workflow management systems. In *Parallel and Distributed Processing, 1996. Eighth IEEE Symposium on*, pp. 581–588. IEEE.

Andrews, T., F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, et al. (2003). Business process execution language for web services.

Barros, A., M. Dumas, et A. H. Ter Hofstede (2005). Service interaction patterns. In *Business Process Management*, pp. 302–318. Springer.

Bhiri, S., C. Godart, et O. Perrin (2006). Transactional patterns for reliable web services compositions. In *Proceedings of the 6th international conference on Web engineering*, ICWE '06, New York, NY, USA, pp. 137–144. ACM.

Bhiri, S., O. Perrin, et C. Godart (2005). Ensuring required failure atomicity of composite web services. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, New York, NY, USA, pp. 138–147. ACM.

Bunting, D., M. Chapman, O. Hurley, M. Little, J. Mischkinsky, E. Newcomer, J. Webber, et K. Swenson (2003). Web services transaction management (ws-txm) version 1.0. *Arjuna, Fujitsu, IONA, Oracle, and Sun*.

Elmagarmid, A. K. (1991). Transaction models for advanced database applications.

Freund, R. W., H. T. Freund, F. Leymann, I. Robinson, et T. Storey (2005). Web services business activity framework (ws-businessactivity).

Graiet, M., I. Abbassi, L. Hamel, M. T. Bhiri, M. Kmimech, et W. Gaaloul (2013). Event-b based approach for verifying dynamic composite service transactional behavior. In *Proceedings of the 2013 IEEE 20th International Conference on Web Services*, ICWS '13, Washington, DC, USA, pp. 251–259. IEEE Computer Society.

Imed, A., G. Mohamed, et C. Eric (2013). Vérification formelle des services web composés transactionnels dynamiques. In *7ème édition de la Conférence francophone sur les Architectures Logicielles (CAL 2013)*.

Jablonski, S. et C. Bussler (1996). Workflow management : modeling concepts, architecture and implementation.

Jureta, I. J., S. Faulkner, Y. Achbany, et M. Saerens (2007). Dynamic web service composition within a service-oriented architecture. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pp. 304–311. IEEE.

Kavantzas, N., D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, et C. Barreto (2005). Web services choreography description language version 1.0. *W3C candidate recommendation 9*.

Lam, V. S. W. (2008). Dynamic workflow patterns. In *Enterprise Information Systems and Web Technologies*, pp. 160–166.

Mustafa, F. et T. L. McCluskey (2009). Dynamic web service composition. In *Proc. Int. Conf. Computer Engineering and Technology ICCET '09*, Volume 2, pp. 463–467.

Russell, N., A. H. Ter Hofstede, D. Edmond, et W. M. van der Aalst (2005a). Workflow data patterns : Identification, representation and tool support. In *Conceptual Modeling–ER 2005*, pp. 353–368. Springer.

Russell, N., A. H. Ter Hofstede, et N. Mulyar (2006). Workflow controlflow patterns : A revised view.

Russell, N., W. M. van der Aalst, A. H. ter Hofstede, et D. Edmond (2005b). Workflow resource patterns : Identification, representation and tool support. In *Advanced Information Systems Engineering*, pp. 216–232. Springer.

Snell, J., D. Tidwell, et P. Kulchenko (2009). *Programming Web services with SOAP*. O'reilly.

Tidwell, D., J. Snell, et P. Kulchenko (2001). *Programming Web Services with SOAP*. O'Reilly.

van Der Aalst, W. M., A. H. Ter Hofstede, B. Kiepuszewski, et A. P. Barros (2003). Workflow patterns. *Distributed and parallel databases 14*(1), 5–51.

W. M. P. van der Aalst, A. P. Barros, A. H. M. t. H. et B. Kiepuszewski. (2000). Advanced workflow patterns. In *O. Etzion and Peter Scheuermann, editors, 5th IFCIS Int. Conf. on Cooperative Information Systems (CoopIS'00)*, Volume number 1901 in LNCS, pp. 18–29.

Patrons transactionnels dynamiques pour des services composés fiables et flexibles

Summary

In this paper, we propose a solution to specify a flexible and reliable dynamic services compositions. So, we introduce a new paradigm, called dynamic transactional pattern. This paradigm is a convergence concept between dynamic workflow patterns and advanced transactional model. It can be seen as dynamic coordination and as a structural transaction. Thus, it combines dynamic Control-Flow flexibility and transactional processing reliability. A set of transactional patterns instances are dynamically connected together in order to define a dynamic composite Web service. We use a set of rules to ensure control and transactional consistency between patterns inside a dynamic services composition.