Opération QSL : COWS

Contraintes pour la Composition de services Web

Responsables :
Olivier Perrin, Laurent Vigneron
LORIA
BP 259, 54506 Vandoeuvre-les-Nancy Cedex

Participants :
Claude Godart, (PR) LORIA/UHP
Olivier Perrin, (MCF) LORIA/Nancy 2
Silvio Ranise, (CR) LORIA/INRIA-Lorraine
Christophe Ringeissen, (CR) LORIA/INRIA-Lorraine
Michaël Rusinowitch, (DR) LORIA/INRIA-Lorraine
Laurent Vigneron (MCF) LORIA/Nancy 2

Laboratoire : LORIA
Durée : 2 ans
Résumé :

Les architectures orientées services sont souvent critiquées pour leur traitement purement syntaxique des concepts de composition, coordination, et surveillance – concernant les couches hautes de l'architecture – et qui rend leur mise en oeuvre très complexe.

Pour pallier ces difficultés, les services Web sémantiques ne se contentent pas de définitions syntaxiques, mais apportent plus d'informations pour permettre de composer, de coordonner, et de surveiller plusieurs services Web.

Ce projet vise à étudier l'utilisation du raisonnement par contraintes et des techniques de preuves pour les services Web. Nos objectifs sont multiples. Un premier objectif est de spécifier dans un cadre formel la composition de services Web. Un deuxième concerne la coordination d'un ensemble de services, pour garantir que l'exécution est conforme à ce qui est attendu par toutes les parties. Enfin, un troisième objectif concerne la conception de mécanismes permettant de surveiller la manière dont une composition précédemment spécifiée

s'exécute, et de savoir réagir lorsque survient une exception (ou un comportement anormal). Il est clair que les objectifs sont étroitement liés, et que nous visons la définition d'un modèle global permettant de composer, de coordonner, et de surveiller.
Mots-clés : contraintes, raisonnement automatique, preuves, services Web.

1 Contexte de la recherche

Actuellement, les architectures orientées services souffrent d'une critique récurrente qui concerne l'aspect uniquement syntaxique des concepts. Si tout le monde s'accorde sur l'intérêt d'utiliser le protocole SOAP comme protocole de communication, la composition, la coordination, et la surveillance – c'est-à-dire les couches hautes de l'architecture – deviennent complexes à mettre en place lorsqu'elles sont vues uniquement d'un point essentiellement syntaxique.

Dans une telle architecture, les échanges de messages jouent un rôle primordial, et l'objectif est alors d'assurer des interactions et des échanges de messages fiables. La fiabilité regroupe deux dimensions : la première a trait au point de vue transactionnel, alors que la seconde concerne le point de vue sécurité. Les échanges de messages doivent donc être exécutés dans un ordre donné pour atteindre le résultat visé (la réservation d'un voyage, avec location de véhicule, et réservation d'hôtel est l'application la plus souvent exposée, mais on retrouve des procédés plus complexes dans les organisations virtuelles par exemple). De même, lorsqu'un ordre d'achat a été validé, l'acheteur s'attend à recevoir son dû, et le vendeur le paiement. Enfin, les apsects sécurité (fraude, non-répudiation, ...) sont importants.

Pour orchestrer les messages, ceux-ci peuvent respecter un procédé. Comme WSDL [13, 25] ne permet pas de définir un tel procédé, différents langages ont vu le jour pour supporter cette modélisation : WSBPEL [1], ebXML [3, 2], ou WS-CDL [12]. Tous ces langages de définition de procédés permettent la définition de patrons d'échanges de messages (Message Exchange Patterns), en s'inspirant fortement des recherches sur les systèmes de workflow. Ainsi, la séquence des échanges de messages est définie. Cependant, à aucun moment, les aspects sémantiques sont abordés. Or, dans le contexte des services Web, les aspects dynamicité (service à la demande et remplacement à la volée d'un service par un autre ayant les mêmes fonctionnalités) et vérification/correction (livraison et/ou paiement) sont primordiaux. Par exemple, on ne peut pas permettre qu'après l'exécution d'un message spécifique, il est impossible de revenir en arrière, c'est-à-dire d'annuler facilement la transaction. Cela signifie que, bien que cela puisse être défini dans le procédé, les services Web actuels ne permettent pas de gérer seuls cette situation, ce qui peut se révéler gênant puisque cela crée un contentieux entre le demandeur et le fournisseur du service.

Pour essayer de pallier ces difficultés, les services Web sémantiques ne se contentent pas de définitions syntaxiques, mais apportent des informations permettant de composer, de coordonner, et de surveiller plusieurs services Web.

1.1 Qualification des partenaires

Le projet regroupe des chercheurs ayant les spécialités suivantes : travail coopératif pour le Web sémantique, raisonnement par contraintes et par règles, coopération de solveurs, preuves de protocoles, procédures de décision pour la preuve. Du côté LORIA, le projet regroupe des membres des équipes ECOO et CASSIS. Du côté de nos collaborateurs externes, on trouve des chercheurs souhaitant adopter une approche orientée services Web pour la collaboration de solveurs.

Verrous scientifiques La technologie des services Web est amenée à croître ces prochaines années. Cette technologie reposant sur les échanges de messages, les verrous actuels se situent au niveau de la fiabilité. Il n'existe pas de propositions concernant la fiabilité pour les aspects transactionnels, et les propositions telles que WS-Security ou WS-Trust ne prennent pas en compte la non-répudiation par exemple.

Pour les protéger des accès illégaux, il est impératif d'étudier les problèmes de sécurité sous-jacents, notamment la définition et le respect des politiques de sécurité. La modélisation des politiques de sécurité

pour les services Web est encore un problème largement ouvert. De plus, il est essentiel de s'assurer que les échanges de messages lors de protocoles de coordination ne peuvent pas être exploités de façon illégale.

Méthodologie Pour travailler sur les problèmes de fiabilité et de sécurité des services Web, nous souhaitons d'une part utiliser le raisonnement par contraintes et les techniques de preuves développées pour la vérification de protocoles (cryptographiques) et la vérification de composants logiciels, et d'autre part d'utiliser les travaux d'ECOO sur les aspects transactionnels pour appliquer une méthode similaire pour la sécurité des échanges de messages.

2 Objectifs de cette opération

Ce projet vise à étudier l'utilisation du raisonnement par contraintes pour la conception sûre de services Web. Nos objectifs sont multiples. Un premier objectif est de spécifier formellement la composition de services Web. Un deuxième concerne la coordination d'un ensemble de services, en garantissant que l'exécution est sûre, en étant conforme à ce qui est attendu par toutes les parties. Enfin, un troisième objectif concerne les mécanismes permettant de surveiller la manière dont une composition précédemment spécifiée s'exécute, et de savoir réagir lorsque survient une exception. Il est clair que les objectifs sont étroitement liés, et que nous visons la définition d'un modèle permettant de composer, de coordonner, et de surveiller.

2.1 Composition

Composer plusieurs services Web requiert plusieurs tâches : la description et la découverte des services, l'orchestration et la surveillance, et la coordination de l'échange des différents messages (et des informations). On obtient alors, de manière automatique ou manuelle, un schéma de composition. Une fois que le schéma de la composition est déterminé, on peut l'exécuter, on parle alors d'orchestration.

Plusieurs modèles existent déjà, mais aucun ne s'est vraiment imposé. En particulier :

- WSBPEL [1], les Mealy machines : orientés orchestration et échange,
- OWL-S [29] : orienté activités et découverte,
- le modèle Romain [20] : orienté activités, orchestration.

La composition de services peut être abordée selon plusieurs disciplines. Le premier domaine intéressant est celui des workflows, et de l'algèbre des processus. Un autre domaine concerne la planification (IA) qui permet de construire une composition en spécifiant son objectif, et en laissant à un algorithme de planification le soin de réaliser la meilleure composition possible étant donnée un ensemble de services. C'est la voie choisie au CMU avec OWL-S, et par l'université de Trento [33]. Une autre possibilité est l'utilisation de logiques, comme par exemple la logique temporelle [24], la logique de description [19], le calcul de situation, ou encore la logique propositionnelle. Enfin, les bases de données permettent d'apporter tout un ensemble de modèles pour tout ce qui concerne les transactions.

Les perspective que nous visons à travers cette action sont les suivantes :

- utiliser le raisonnement par contraintes pour améliorer et guider la composition de services,
- modéliser un schéma de composition pour permettre de vérifier a priori ses qualités, de découvrir ses défauts (par exemple les problèmes liés aux aspects transactionnels) et vérifier le respect de politiques de sécurité.

En terme de modèle, les différentes approches possibles pour la composition sont basées sur les automates (orientée messages ou activités), sur la logique, ou sur les contraintes (spécification du comportement attendu du service). Actuellement, l'équipe ECOO utilise une modélisation à base de diagrammes à changement d'états, et nous avons proposé un algorithme qui permet de vérifier si une composition respecte un certain nombre de propriétés du point de vue transactionnel [22]. Dans le cadre de cette proposition, nous souhaitons étudier une modélisation à base de contraintes ayant la capacité de raisonner sur les aspects transactionnels, avec la possibilité d'aborder d'autres aspects : équité, sécurité,... Nous envisageons d'étudier comment la réalisation d'une composition de services basée sur les échanges de messages peut être vue comme un problème de recherche de solution pour un système de contraintes.

2.2 Coordination/Exécution

Si la composition peut être vue comme la possibilité de créer un nouveau service à partir de services existants, la coordination (aussi appelée chorégraphie) est plutôt orientée vers la modélisation des échanges entre plusieurs services, permettant de vérifier la correction des protocoles de coordination entre chaque service (soit de manière centralisée, soit en point à point). La coordination définit le séquencement et les conditions sur les échanges de messages qui permettent aux services qui coopèrent d'atteindre un objectif. Ainsi, la coordination de plusieurs services permet de définir le protocole d'échange de messages, les interfaces, le séquencement, et la logique associée. Le protocole est constitué d'un ensemble de règles définissant les messages, leur séquencement, les échanges de données,... Une conversation est alors vue comme une instance d'un protocole [23].

Dans le contexte des services Web, définir un protocole est important puisque les services Web sont orientés, de par leur nature vers le mode conversationnel, et que le faible couplage entre plusieurs services nécessite que chaque service possède des spécifications précises en ce qui concerne sa contribution à l'exécution d'une tâche. Ainsi, avoir un protocole, qui plus est une spécification du protocole, permet d'envisager une analyse formelle de son déroulement, et d'effectuer des comparaisons, des requêtes,... Les caractéristiques d'un "bon" protocole sont : un modèle formel, la définition des états, la manipulation des données, un caractère déterministe, un comportement capable de supporter le temps, la gestion des exceptions, une implantation facile, l'intégration avec les standards existants, et son adéquation avec les services Web (puissant mais simple).

Les formalismes actuels pour modéliser les protocoles sont les diagrammes de séquences, les automates à états finis, les modèles basés sur les procédés, et les modèles à base de règles. Formaliser permet de répondre aux questions suivantes : compatibilité entre services et protocoles (totale, partielle), possibilité de remplacer un service par un autre (équivalence, subsomption, intersection, différence, substitution statique grâce à un intermédiaire, substitution dynamique grâce à un adaptateur), conformité de services par rapport à un protocole fixé, ou d'un protocole par rapport à un ensemble de services, adaptation de service/protocole,...

Les trois points difficiles auxquels nous comptons nous intéresser sont : (1) comment vérifier statiquement les propriétés transactionnelles et les politiques de sécurité le protocole de coordination d'un schéma de composition (tout en conservant/garantissant l'autonomie de chaque service), (2) comment assurer dynamiquement (à l'exécution) le comportement d'un service composé (les propriétés transactionnelles), (3) comment assurer le respect de la politique de sécurité, par exemple la non-répudiation. Les protocoles cryptographiques actuels basés sur les clés ne sont pas adaptés à la non-repudiation (une action qui a été déclenchée est démentie par son émetteur) car les clés sont accessibles à plusieurs parties.

Pour répondre à ces problèmes, nous envisageons d'étudier et d'utiliser des techniques de preuves notamment pour des modèles à base de règles, en nous inspirant de nos travaux développés pour la vérification de protocoles cryptographiques. On pourra également envisager d'utiliser des fonctions incluant un code d'authentification d'un message (Message Authentication Code – MAC), ou des signatures permettant de valider l'origine du message (Proof od Message Origin – PMO) et la réception du message (Proof of Message Receipt – PMR). Cela n'est pas évident à cause de la façon dont fonctionne le protocole de transport de messages SOAP. En effet, un message envoyé avec SOAP peut transiter par plusieurs intermédiaires, ce qui rend non triviale l'authentification d'un message entre son émetteur et son récepteur.

2.3 Surveillance

En ce qui concerne la surveillance, nous souhaitons nous intéresser plus particulièrement à tracer l'exécution, sans remettre en cause l'autonomie de chaque partenaire, en profitant au maximum des bénéfices des architectures orientés services concernant la surveillance. Pour cela, nous avons déjà initié un certain nombre de travaux qui utilise le calcul d'évènement. Ce calcul est facilement exploitable par une architecture orientée services, dans laquelle on peut connaitre l'état des services grâce aux messages (évènements) qu'ils engendrent. Avoir un tel formalisme nous a permis de vérifier certaines propriétés liées au protocole de coordination d'un ensemble de services. Nous envisageons d'automatiser ce type de vérifications grâce à un codage du calcul d'évènement [30, 31] en logique du premier ordre et à l'utilisation d'un système de preuve

(du premier ordre), en l'occurrence le système haRVey [27] développé dans l'équipe CASSIS. On pourrait ainsi imaginer que haRVey assure un service de supervision fonctionnant en "ligne" pour vérifier chaque étape de l'exécution.

3 Calendrier

Les deux premières directions de recherche nécessiteront une meilleure compréhension de la notion de politique de sécurité pour les services Web. Après la définition d'un modèle de services Web respectant une politique de sécurité, les travaux concernant la composition et la coordination pourront être menés en parallèle. Concernant la surveillance, les travaux relatifs à l'utilisation d'un calcul d'évènement pourront démarrer plus rapidement sans attendre la modélisation des politiques de sécurité pour les services Web.

4 Contributions à la plateforme QSL

Les outils développés dans le cadre de cette opération seront rendus disponibles sur la plateforme. L'approche orientée services sera sans nul doute profitable à la plateforme. On pourra par exemple l'appliquer à coordonner de manière sûre plusieurs prouveurs de la plateforme vus comme des services.

5 Résultats escomptés au terme de l'action

1. Rapports de recherche, articles dans des conférences et revues
2. Un modèle de services Web
3. Des outils de résolution et de preuve pour supporter le modèle de services Web et disponibles sur la plateforme.

6 Collaborations

Cette demande d'opération QSL est liée à des collaborations que nous entretenons sur ce sujet. Des demandes voisines d'ARA (Action de Recherche Amont) et de coopération INRIA-CONICYT (Chili) sont en cours d'évaluation. On trouvera ci-dessous la liste de nos principaux partenaires hors LORIA avec lesquels nous avons l'intention de travailler sur les thèmes de l'opération.

UTFSM Valparaiso, Chili	Eric Monfroy/ Carlos Castro
IRIT Toulouse, France	Yannick Chevalier
LIFC Besancon, France	Fabrice Bouquet

Références

[1] BPEL. http ://www.oasis-open.org.

[2] ebXML. http ://www.oasis-open.org.

[3] Electronic Business using eXtensible Markup Language. http ://www.ebxml.org.

[4] Organization for the Advancement of Structured Information Standards. http ://www.oasis-open.org/home/index.php.

[5] OWL. http ://www.w3.org/2004/OWL.

[6] PSL. http ://www.tc184-sc4.org.

[7] RDF-S. http ://www.w3.org/RDF.

[8] Universal Discovery, Description and Integration. http ://www.uddi.org.

[9] Web Services Composite Application Framework. http ://developers.sun.com/techtopics/webservices/wscaf.

[10] Web Services Coordination (WS-C). http ://www-106.ibm.com/developerworks/library/ws-coor/.

[11] Web Services Transaction (WS-T). http ://www-106.ibm.com/developerworks/webservices/library/ws-transpec/.

[12] WS-CDL. http ://www.w3.org/2002/ws/chor/.

[13] WSDL. http ://www.w3.org/TR/wsdl.

[14] WSMO. http ://www.wsmo.org.

[15] XML Schema. http ://www.w3.org.

[16] Web Services Conversation Language (WSCL) 1.0. W3C Note, March 2002. http ://www.w3.org/TR/wscl10/.

[17] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services. Concepts, Architectures and Applications*. Springer-Verlag, 2004.

[18] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatteand I. Trickovic, and S. Weerawarana. Business process execution language for web services (bpel4ws) -version 1.1. Technical report, IBM, 2004. http ://www-106.ibm.com/developerworks/library/ws-bpe.

[19] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider. *The Description Logic Handbook : Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[20] D. Berardi. *Automatic Service Composition. Models, Techniques and Tools*. Phd thesis, Universita di Roma La Sapienza, 2005.

[21] D. Berardi, R. Hull, M. Gruninger, and S. McIlraith. Towards a First-Order Ontology for Semantic Web Services. In *Proc. W3C International Workshop on Constraints and Capabilities for Web Services (WS-CC)*. http ://www.w3.org/2004/06/ws-cc-cfp.html.

[22] S. Bhiri, O. Perrin, and C. Godart. Ensuring Required Failure Atomicity of Composite Web Services. In *Proc. of WWW'2005*, 2005.

[23] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification : a new approach to design and analysis of e-service composition. In *Proc. of WWW'2003*, pages 403–410, 2003.

[24] D. Calvanese, G. De Giacomo, and M. Y. Vardi. Reasoning about Actions and Planning in LTL Action Theories. In *Proc.of KR*, pages 593–602, 2002.

[25] R. Chinnici, M. Gudgin, J.J. Moreau, J. Schlimmer, and S. Weerawarana. Web services description language (wsdl) 2.0. Working draft, W3C, 2003. Available on line : http ://www.w3.org/TR/wsdl20.

[26] A. Dan and D. Davis et al. Web services on demand : Wsla-driven automated management. *IBM Systems Journal*, 43(1), 2004.

[27] D. Déharbe and S. Ranise. Light-Weight Theorem Proving for Debugging and Verifying Units of Code. In IEEE Comp. Soc. Press, editor, *Proc. of the Int. Conf. on Software Engineering and Formal Methods (SEFM03)*, 2003.

[28] N. Kavantzas, D. Burdett, G. Ritzinger, and Y. Lafon. Web services choreography description language (wscdl) version 1.0. Working draft, W3C. http ://www.w3.org/TR/ws-cdl-10/.

[29] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. R. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara. Bringing Semantics to Web Services : The OWL-S Approach. In *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, San Diego, California, USA, 2004.

[30] Erik T. Mueller. Event calculus reasoning through satisfiability. *Journal of Logic and Computation*, 14(5) :703–730, 2004.

[31] Erik T. Mueller and Geoff Sutcliffe. Reasoning in the event calculus using first-order automated theorem proving. In Ingrid Russell and Zdravko Markov, editors, *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference, Clearwater Beach, Florida, USA*, pages 840–841. AAAI Press, 2005.

[32] S. Narayanan and S. A. McIlraith. Analysis and simulation of web services. *Computer Networks*, 42(5) :675–693, 2003.

[33] M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, and P. Traverso. Planning and Monitoring Web Service Composition. In *Proc. of AIMSA*, pages 106–115, 2004.