Laboratoire de Recherche en Informatique

Résumé de thèse

Test symbolique de services Web composites

Présentée par:
Lina Bentakouk
pour l'obtention du
Doctorat du l'université Paris-Sud XI

Jury

Pr. Ana Rosa Cavalli	IT/T l com SudParis	Rapporteur
Pr. Manuel Nunez	Universidad Complutense de Madrid	Rapporteur
Pr. Mohand-Said Hacid	Universit Claude Bernard Lyon 1	Examinateur
Pr. Philippe Dague	Universit Paris-Sud XI	Examinateur
Pr. Marie-Claude Gaudel	Universit Paris-Sud XI	Directrice de th se
Dr. Fatiha Zaïdi	Universit Paris-Sud XI	Co-Encadrant
Dr. Pascal Poizat	Universit Paris-Sud XI	Co-Encadrant

1 d cembre 2007 – 16 d cembre 2011

Contents

CHAPTER 1

INTRODUCTION

L'utilisation des ordinateurs dans notre vie quotidienne est devenu presque instinctive. Afin de satisfaire les besoin des clients, les outils informatique sont de plus en plus complexes. Une telle complexité soulève des questions de confiance autour de tels systèmes.
Trouver les fautes d'un systeème puis les corriger évite une perte de temps et d'argent pour les entreprises, cela diminue aussi les risques portant sur la sécurité des clients qui utilisent ce type système, tel que les équipements médicaux, la construction de véhicules ou d'avions, etc.

Afin d'assurer le bon fonctionnement de ces systèmes complexes, il est important de les vérifier et de les tester. Dans notre travail de thèse nous nous sommes intéressé au test de conformité d'une composition de services Web. Le test de conformité est utilisé pour vérifier si une implementation d'un système est correct par rapport à sa spécification.

Les services Web font partie des nouvelles technologies émergentes. Ce sont des composants accessibles sur le Web grâce à leurs interfaces *Web Service Description Language* (WSDL) [1]. Ils peuvent interagir dynamiquement avec d'autres applications via Internet, et cela de façon indépendante des systèmes d'exploitation ou des langages de programmation utilisés. Les services Web représentent la principale mise en oeuvre de l'Architecture Orientée Service (SOA) [5].
De plus les services Web peuvent être composé sous forme centralisé, appellée orchestration de services, ou sous forme répartie, appellée chorégraphie de services. Le langage Web Services-Business Process Execution Language (WS-BPEL) [9] est un langage normalisé et le plus utilisé pour décrire une composition centralisée de

services Web. Un processus WS-BPEL d´écrit comment plusieurs Web Services peuvent interagir de façon coordonnée pour satisfaire un besoin spécifique. Le langage de description du comportement d'un système est appelé langage de spécification dans le domaine du test et de la vérification.

1.1 Motivation

Notre travail de recherche porte sur le test de conformité de compositions de services Web. Nous nous intéressons au test dans le contexte boite noire, ce qui signifie que nous n'avons pas accès au code décrivant la composition de services. A partir d'une spécification de composition de services Web, nous générons des cas de test qui sont ensuite exécutés sur une orchestration de service (appelée aussi implémentation de services) dans le but de déterminer si cette dernière est conforme à la spécification. Les principales questions et verrous abordés dans notre travail sont :

- **La spécification d'une Composition**: comment devons-nous décrire cette composition ? En d'autres mots quel langage de spécification peut-on utiliser ?
- **Modèle formel**: quel modèle formel utiliser, étendre ou définir pour prendre en considération les caractéristiques du langage de spécification?
- **Approche de Test**: quelle approche de test peut-on utiliser/réutiliser pour ces systèmes ?
- **Gestion des données complexes**: comment générer/exécuter les tests avec des systèmes dépendant des données échangées tout en évitant l'explosion combinatoire d'états ?

1.2 Contributions

Nous avons proposé une approche symbolique de test d'orchestration de services Web en utilisant un solveur de contraintes.
Cette approche consiste en l'utilisation du langage d'orchestration Abstract Business Process Execution Language (ABPEL) comme langage de spécification, et sa traduction dans le modèle des Web-Service Symbolic Transition Systems (WS-STS). Notre deuxième contribution est l'utilisation de l'exécution symbolique du WS-STS modèle combinée à l'utilisation du solveur Z3 pour produire un arbre d'exécution symbolique (SET, Symbolic Execution Tree). Chaque chemin de cet arbre représente un possible cas de test.

Les cas de tests sont ensuite exécutés en interagissant avec l'implémentation. Cette exécution se fait aussi avec l'aide du solveur de contraintes (pour l'instanciation

et la vérification des données échangées). Finalement un verdict de conformité est émis.

Nous avons également proposé l'utilisation d'objectifs de tests pour mieux cibler les cas de tests souhaités, grâce à notre outil qui fait le produit entre le modèle de la spécification et celui des objectifs de test. Le premier solveur que nous avons utilisé (UMLtoCSP [4]) était limité, car on était obligé à chaque fois de parcourir l'arbre d'exécution symbolique jusqu'au feuilles avant d'envoyer les contraintes au solveur, pour palier à ce problème nous avons étudier l'utilisation du solveur SMT Z3 [8] afin de résoudre des contraintes sur des données de type arbre à feuilles entières. L'outil qui permet la traduction de ces contraintes, l'appel au solveur Z3 et l'analyse du retour de réponse du solveur est en cours de finalisation.

1.3 Publications

- Lina Bentakouk, Pascal Poizat and Fatiha Zaïdi. A Formal Framework for Service Orchestration Testing Based on Symbolic Transition Systems. In Proceedings of the 21st IFIP WG 6.1 International Conference on Testing of Software and Communication Systems and 9th International FATES Workshop (TESTCOM '09/FATES '09). LNCS 5826, pages 16-32, Springer 2009

- Lina Bentakouk, Fayçal Bessayah, Mounir Lallali, Wissam Mallouli and Andrey Sadovykh. A Framework for Modeling and Testing of Web Services Orchestration. In MDA4ServiceCloud 2010 - The 4th workshop on Modeling, Design, and Analysis for the Service Cloud, Paris, France. 2010.

- Lina Bentakouk, Pascal Poizat and Fatiha Zaïdi. Checking the Behavioral Conformance of Web Services with Symbolic Testing and an SMT Solver. In TAP'11 Proceedings of the 5th international conference on Tests and proofs In TAP'2011. LNCS 6706, pages 33-50 Springer 2011.

CHAPTER 2

Approche Symbolique de Test de Conformité pour une Composition de Services

2.1 Description de notre approche symbolique de test

Dans cette partie nous présentons l'aspect théorique de notre approche symbolique pour le test d'une orchestration de Web services. A partir d'une spécification de composition de service Web, qui permet de décrire les fonctionnalités de cette composition, nous appliquons notre approche de test comme décrit dans la Figure 2.1

Cette approche consiste en trois étapes: la modélisation, la génération de cas de test puis l'exécution des cas de tests.

La modélisation - À partir d'une spécification de service composite, plus précisément une orchestration de services, décrite en *Abstract Business Process Execution Language* (ABPEL), on produit un modèle nommé *Web Service Symbolic Transition Systems* (WS-STS) qui reprend de façon formelle et sans ambiguïté le comportement décrit par la spécification, grâce a nos règles de transformations. Nous fournissons également la possibilité de se focaliser sur un certain comportement à tester en utilisant des objectifs de teste (TP, *Test Purposes*). Un TP est fournie sous la forme d'un modèle formel WS-STS.

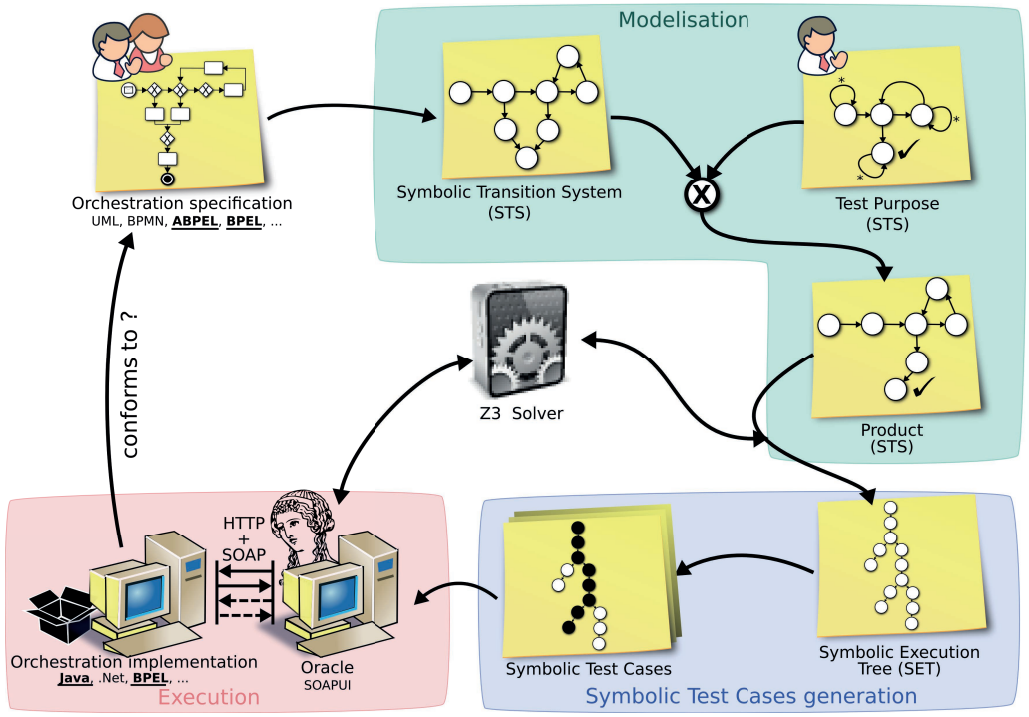

Figure 2.1: Vue globale de notre approche symbolique de test

Afin d'inclure l'objectif de test au mod le de sp cification, un produit est calcul entre les deux pr c dent mod le. Le produit fournie un nouveau mod le de test, ce dernier d crit la sp cification mais aussi pend en consid ration l'objectif de test.

La génération de cas de test - Par la suite, un arbre d'ex cution symbolique (SET, *Symbolic Execution Tree*) est calcul partir du mod le de la sp cification ou du mod le produit entre le mod le de sp cification et du mod le de l'objectif de test. Pour ce faire nous avons d fini des r gles de construction des noeuds symboliques en sp cifiant comment traiter les expressions XPath. L'arbre repr sente la s mantique du mod le obtenu en utilisant la technique de l'Ex cution Symbolique (SE, *Symbolic Execution*). Cette technique nous permet d' viter les probl mes d'explosion de l'espace d' tat en pr sence de types de donn es illimit es, qu'utilis dans ABPEL. Chaque chemin de l'arbre SET correspond a un potentiel cas de test. Il est possible que la g n ration du SET soit infinie, d l'ex cution d'une boucle dans le mod le par exemple. Pour limiter le d ploiement infinie de l'arbre nous proposons l'utilisation de deux crit res d'arr t: (1) limiter la g n ration de l'arbre une profondeur d finie, (2) utiliser le crit re d'inclusion qui consiste stopper la g n ration de d'une branche de l'arbre qui d crit un comportement r p titif.

L'exécution - Enfin, les cas de tests sont exécuter contre l'implémentation de service avec l'utilisation du solveur SMT Z3. Ce dernier est utilisé pour (1) fournir les données d'entrée soumis à l'implémentaion, (2) vérifier la validité des données retournées par cette implémentation. Finalement, un verdict portant sur la conformité de l'implémentation par rapport à sa description est émis.

2.2 Outils utilisés

Notre approche est soutenue par une chaîne d'outil associée à chaque étape comme décrit dans la Figure 2.2.

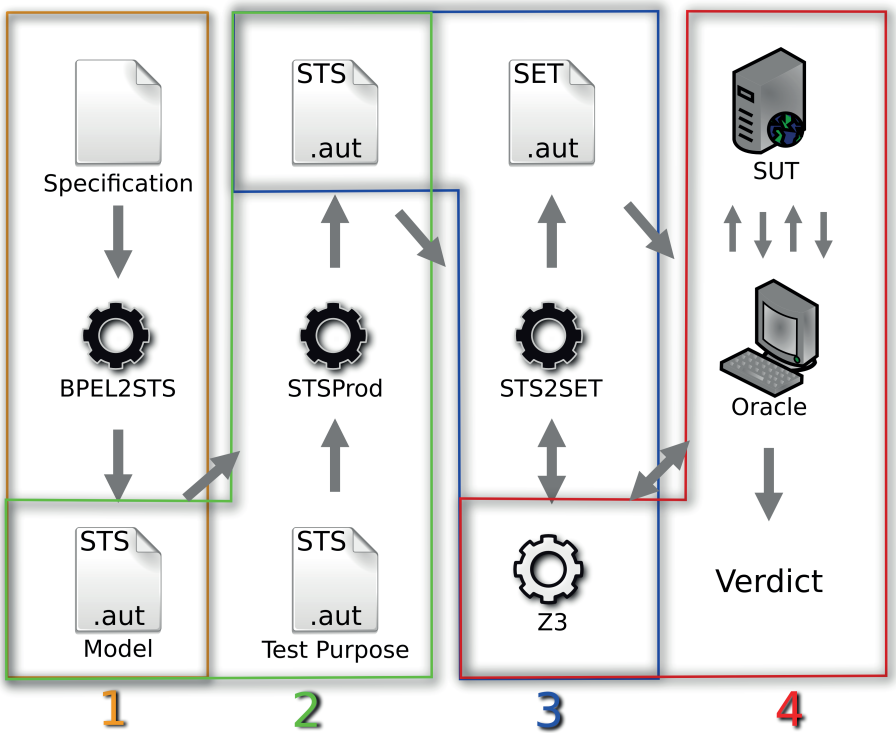

Figure 2.2: Vue globale de la chaîne d'outil

Nous proposons une approche globale de bout en bout pour le test de conformité d'une orchestration de services. Pour ce faire, nous commençons avec la description de l'orchestration en ABPEL, à partir de laquelle nous produisons le modèle formel WS-STS. Ce modèle est obtenu grâce a nos règles de transformation décrits comme algèbre de processus, le but est de donner une sémantique opérationnel à la spécification, afin de générer les tests. Cette étape est exécutée grâce a l'outil *BPEL2STS*.

L'outil *STSProd* permet de calculer le modèle produit qui représente la prise en compte de l'objectif de test avec le modèle de la spécification.

Le modèle de la spécification ou le modèle de produit, dans le cas ou on a un objectif de test, est utilisé par l'outil *STS2SET* ainsi que le solveur SMT Z3 afin de générer l'arbre d'exécution symbolique (SET). Chaque branche du SET représente un cas de test potentiel.

Enfin, nous avons donné un algorithme d'exécution des tests qui interagit avec le solveur Z3, et qui nous permet d'émettre un verdict sur la conformité de l'implémentation par rapport à sa spécification. Actuellement cette partie est exécutée manuellement en utilisant (1) l'outil *soapUI* [2] afin d'intéragir avec le service composite et (2) en utilisant le SMT solveur Z3 afin de fournir les données en entrée puis de vérifier la satisfiabilité des réponse retourné par le service composite.

CHAPTER 3

Conclusion et Perspectives

Nous nous sommes intéressés dans le test de conformité d'une implémentation d'une orchestration de services Web par rapport à sa spécification. Le manuscrit de thèse présente notre approche symbolique proposée pour le test de conformité d'une orchestration de service décrite en *Abstract Business Process Execution Language* (ABPEL).

Les perspectives pour notre travail peuvent être divisé en perspectives de recherche et perspectives de développement d'outils.
Parmi les perspectives de recherche, il serait intéressant de définir un langage de spécification pour les objectifs de test qui soit plus simple que le modèle formel du WS-STS. Une autre perspectives serai de prendre en considération le langage *UML4SOA* [7] ou le langage *SRML* [3] afin de spécifier une composition de services. Dans notre approche nous nous somme focaliser sur la composition centraliser de services, une autre perspectives serait d'adapter et de modifier notre approche pour une composition répartie de services.

La première perspectives de développement d'outils serait d'automatiser le processus d'exécution des cas de test sur le service composite. Une autre amélioration de notre approche serait l'incorporation de solveur pour les variables dont le type est String tel que le solveur HAMPI [6].

Bibliography

[1] Web services description language (wsdl). http://www.w3.org/TR/wsdl. URL http://www.w3.org/TR/wsdl. 1

[2] soapui tool. URL http://www.soapui.org/. 8

[3] Sensoria reference modeling language. URL http://www.cs.le.ac.uk/srml/. 9

[4] Jordi Cabot, Robert Clarisó, and Daniel Riera. UMLtoCSP: a Tool for the Formal Verification of UML/OCL Models using Constraint Programming. In *Proc. of ASE*, 2007. 3

[5] Thomas Erl. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services.* Prentice Hall, April 2004. 1

[6] Adam Kiezun, Vijay Ganesh, Philip J. Guo, Pieter Hooimeijer, and Michael D. Ernst. Hampi: A solver for string constraints, 2009. 9

[7] Philip Mayer, Andreas Schroeder, and Nora Koch. A model-driven approach to service orchestration. In *Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 2*, pages 533–536, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3283-7-02. doi: 10.1109/SCC.2008.91. URL http://portal.acm.org/citation.cfm?id=1443230.1444290. 9

[8] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *Proc. of TACAS*, 2008. 3

[9] Oasis. *Web Services Business Process Execution Language (WSBPEL) Version 2.0*, April 2007. URL http://docs.oasis-open.org/wsbpel/2.0/Primer/wsbpel-v2.0-Primer.html. 1