GENERACIÓN DE CASOS DE PRUEBA PARA COMPOSICIONES DE SERVICIOS WEB ESPECIFICADAS EN BPEL

José García-Fanjul, Javier Tuya y Claudio de la Riva

Departamento de Informática
Universidad de Oviedo
Campus de Viesques, S/N. 33204 – Gijón (España)
e-mail: {jgfanjul,tuya,claudio}@uniovi.es

Palabras clave: composiciones de servicios web, comprobación de modelos, pruebas basadas en modelos, pruebas del software

Resumen. *La prueba de software basado en servicios web resulta especialmente compleja, dada su naturaleza distribuida y su comportamiento asíncrono. Sin embargo, la investigación en este campo es, aún, escasa. En este artículo, se propone un enfoque nuevo para la prueba de composiciones de servicios web especificadas en el lenguaje estándar BPEL. En concreto, se utilizará un comprobador de modelos para generar automáticamente casos de prueba para estas composiciones. También se propondrá la utilización de criterios de suficiencia con el objetivo de seleccionar, sistemáticamente, los casos de prueba. Se describen, asimismo, resultados preliminares que se han obtenido aplicando un criterio de cobertura de transiciones.*

1. INTRODUCCIÓN

Los servicios web están pasando a ser la elección natural para implementar software distribuido. La definición de interfaces a través de datos en XML hace que la composición de servicios web permita la implementación de procesos de negocio interoperables. Ésta es una de las causas principales de que la inversión en este tipo de software, a nivel mundial, se haya incrementado drásticamente en los últimos años. En concreto, según los estudios realizados por la consultora IDC [1], dicha inversión se ha duplicado del año 2003 al 2004, alcanzando la cifra de 2,3 miles de millones de dólares. Para el año 2009 se prevé que la cantidad invertida sea de 15 miles de millones de dólares. Esta buena aceptación por parte de la industria se ha traducido en el desarrollo de lenguajes estándar para la especificación de composiciones de servicios web, como BPEL [2].

Por otro lado, las investigaciones han empezado a centrarse en el modo de probar este tipo de software. Debe determinarse en qué medida los procesos de prueba válidos para otro tipo de software pueden ser aplicados a servicios web, y qué investigaciones específicas para este

tipo de sistemas deben ser realizadas. Canfora y Di Penta [3] y Zhang y Zhang [4] han identificado varias peculiaridades de los servicios web que influyen en la forma en que debe probarse este tipo de software. Algunas de ellas son las siguientes:

1. Las pruebas sobre servicios web deben ejecutarse remotamente, por lo que debe asumirse un sobrecoste asociado al mantenimiento de la propia conexión y a condiciones impredecibles como el tráfico y la seguridad.
2. El diseño de los casos de prueba debe poder abordarse a partir de la información limitada que, habitualmente, está disponible sobre el comportamiento de un servicio web. En concreto, es muy frecuente que sólo se disponga de datos sobre el interfaz de dicho software.

Teniendo en cuenta dichas peculiaridades, en este artículo se propone un nuevo enfoque para la generación de casos de prueba para composiciones de servicios web. El trabajo relacionado se revisará en la Sección 2. Después, en la Sección 3, se especificará nuestra propuesta. Para finalizar, se resumen las principales contribuciones en la Sección 4.

2. TRABAJO RELACIONADO

La investigación sobre procesos de verificación y validación aplicados sobre composiciones de servicios web puede clasificarse en dos categorías: trabajos que describen enfoques basados en el uso de técnicas de verificación formal y otros que utilizan técnicas preexistentes de pruebas.

La mayor parte de la investigación en este campo se ha dirigido al empleo de técnicas de verificación formal. Su objetivo es decidir si la composición que se estudia cumple con ciertas propiedades. Fu y otros [5] utilizan el comprobador de modelos SPIN [6] para verificar formalmente composiciones de servicios web especificadas en BPEL. Pese a que en este artículo se utiliza la misma herramienta, nuestro objetivo es la detección de errores a través de la generación de casos de prueba seleccionados sistemáticamente, y no la verificación formal de la composición. En la misma línea de trabajo, Foster y otros [7] usan Finite State Processes (FSP) para modelar composiciones de servicios web que verifican formalmente. Utilizando paradigmas diferentes para los modelos y la verificación, Narayanan y McIlraith [8] proponen completar la especificación del comportamiento de los servicios web con descripciones semánticas en DAML-S, para posteriormente transformarlas en Redes de Petri.

En cuanto a los enfoques orientados a la realización de pruebas, Chun y Offutt [9] y Offutt y Xu [10] describen el uso del análisis de mutantes y la perturbación de datos para la prueba de servicios web. Estas técnicas estarían definidas a nivel de unidad, y por tanto orientadas a la prueba de servicios web individuales, no de su composición. En combinación con técnicas de verificación, Huang y otros [11] describen un método para probar composiciones de servicios web. En dicho trabajo, se propone especificar explícitamente el comportamiento de los servicios web (utilizando OWL-S) y definir las propiedades deseadas manualmente. A continuación, proponen la utilización de comprobadores de modelos para determinar si las propiedades se cumplen.

Varios de los trabajos que se acaban de mencionar (como [5], [8] y [11]) se basan en la anotación explícita del comportamiento de los servicios web. Creemos, por tanto, que es

necesario continuar investigando en métodos de prueba que dependan exclusivamente de la información contenida en especificaciones basadas en estándares industriales de amplia aceptación, como BPEL.

3. MÉTODO PROPUESTO

El método propuesto consiste en la generación de casos de prueba a partir de un modelo de la composición de servicios web. Dicho modelo se obtendrá a partir de una especificación BPEL y se expresará en el lenguaje PROMELA, que es el lenguaje de entrada del comprobador de modelos SPIN. Para generar los casos de prueba, se adaptará una técnica propuesta por Ammann et al [12]. En concreto, se obtendrá un caso de prueba para una determinada condición C proporcionando al comprobador un modelo de la composición BPEL y una fórmula LTL que especifique que C nunca se cumple. Durante su ejecución, SPIN realiza una búsqueda entre todos los posibles estados en los que puede estar el modelo, y comprueba si las propiedades se cumplen. La salida obtenida de la herramienta será, entonces, un contraejemplo en el que el proceso de negocio cumple C. Dicho contraejemplo puede ser transformado en un caso de prueba, puesto que describe una ejecución de la composición de servicios web en la cual se cumple la condición de prueba deseada. La Figura 1 ilustra el método propuesto, que se compone de cuatro fases:

- Fase 1.- Transformar la especificación BPEL a PROMELA. En primer lugar, el comportamiento del proceso de negocio se transforma a PROMELA. Para definir casos de prueba completos, es necesario modelar también el comportamiento externo de los diferentes servicios web que participan en la composición. La especificación BPEL puede no incluir, directamente, información acerca del comportamiento de los servicios individuales. Por tanto, dicho comportamiento debe ser simulado a partir del interfaz entre el servicio web y el proceso de negocio.
- Fase 2.- Aplicar un criterio de suficiencia. En esta fase se identifican las condiciones de la prueba para cada caso. Se pueden aplicar diferentes criterios de suficiencia, como los descritos por Offutt y otros en [13]. Los criterios guían la instrumentación del código PROMELA, para identificar si la ejecución del modelo cumple las condiciones de la prueba. Además, se construyen fórmulas LTL que expresan la negación de las condiciones de prueba.
- Fase 3.- Ejecutar SPIN. El tercer paso es la ejecución del comprobador de modelos. El contraejemplo que se obtiene es una ejecución del modelo (y por tanto del proceso de negocio) en el que se ejercitan las condiciones de prueba incluidas en la fórmula LTL.

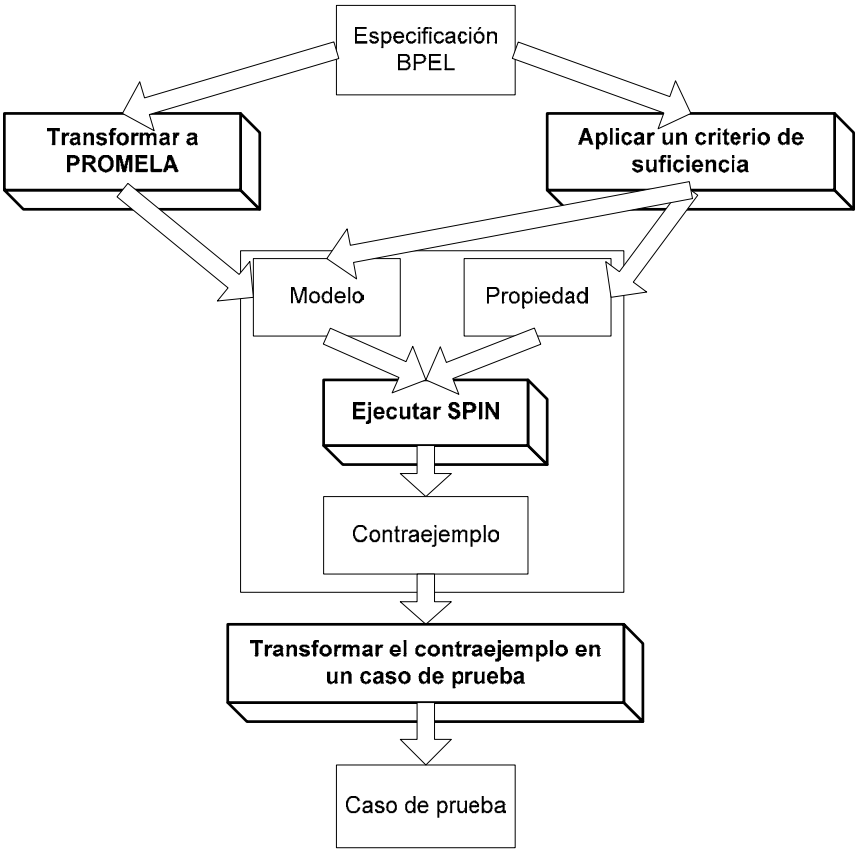

Figura 1. Esquema del método propuesto.

- Fase 4.- Transformar el contraejemplo en un caso de prueba. Por último, para especificar un caso de prueba, se extrae la información relevante a partir del contraejemplo generado por el comprobador. La especificación del caso de prueba incluye las entradas y la salida deseada, ambas expresadas en términos de la información intercambiada entre el proceso de negocio y los servicios web.

En nuestros trabajos preliminares [14] [15], se ha utilizado un criterio de cobertura de transiciones para seleccionar los casos de prueba. Específicamente, dicho criterio establece que el conjunto de casos final debe incluir casos de prueba que recorran todas las transiciones identificadas en la especificación del proceso de negocio. Para ello, en la segunda fase del método, se identifican transiciones en la especificación y se mapean a PROMELA. También se construye una propiedad LTL para cada transición, indicando que dicha transición no se puede ejercitar. Por ejemplo, si en la especificación se identifica una transición denominada `t1`, en PROMELA se utilizará una variable booleana denominada `tran1` que indica si esa transición es ejecutada. Entonces, para generar un caso de prueba que cubra dicha transición, se utilizará la fórmula LTL "`[] !tran1`", que se traduce por "`tran1` siempre es falsa". Al ejecutar el comprobador, éste producirá un contraejemplo

en el que la variable `tran1` es cierta, y por tanto expresa una ejecución de la composición de servicios web en la que se ejercita la transición t^1.
Para generar un conjunto de casos que cumpla el criterio de cobertura de transiciones, el comprobador de modelos debería ejecutarse tantas veces como transiciones sean identificadas. Sin embargo, para reducir el número de casos de prueba, se contabilizan todas las transiciones que ejercita cada contraejemplo. En el primer caso de estudio realizado, sobre el bien conocido ejemplo del "loan approval", el número de casos de prueba obtenidos es el mínimo requerido para dar cobertura de transiciones a la especificación.

4. CONCLUSIONES

En este artículo se describe un método basado en modelos para obtener casos de prueba para composiciones de servicios web especificadas en el lenguaje BPEL. Para ello, inicialmente se obtiene un modelo PROMELA de la composición. A continuación se aplica un criterio de suficiencia, lo que se traduce en la instrumentación del modelo y la obtención de propiedades LTL que especifiquen diferentes condiciones de prueba. En cada ejecución del comprobador, se obtendrá un contraejemplo que debe ser transformado en un caso de prueba. Los trabajos preliminares nos han permitido comprobar la viabilidad del método utilizando, como criterio de suficiencia, un criterio de cobertura de transiciones.
Frente a otros enfoques nuestro método requiere, como única entrada para la generación de los casos, una especificación de la composición en BPEL. Por tanto, los casos de prueba serán independientes de la implementación concreta y, además, el método será directamente aplicable sobre un estándar industrial comúnmente utilizado.
Como líneas de trabajo futuro, se aplicará el método utilizando diferentes criterios de suficiencia. También se debe profundizar en su validación, utilizando mutantes o técnicas de experimentación para medir la efectividad de los casos de prueba generados.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por el Programa Nacional de I+D+I del Ministerio de Educación y Ciencia y fondos FEDER, con los proyectos IN2TEST (TIN2004-06689-C03-02) y RePRIS (TIN2005-24792-E).

REFERENCIAS

[1] IDC, *Research Reports*, URL: http://www.idc.com/.
[2] IBM, *Business Process Execution Language for Web Services version 1.1*, URL: http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.
[3] G. Canfora y M. Di Penta, "Testing services and service-centric systems: Challenges and opportunities", *IT Professional*, Vol. **8**(2), pp.10–17, (2006).
[4] J. Zhang y L.J. Zhang, "Web Services Quality Testing", *International Journal of Web Services Research*, Vol. **2**(2), pp. 1-4, (2005).
[5] X. Fu, T. Bultan y J. Su, *Analysis of Interacting BPEL Web Services*, En

Proceedings of the 13th International World Wide Web Conference, New York - USA, (2004), pp. 621-630.

[6] G.J. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*, Addison-Wesley Professional, 2003.

[7] H. Foster, S. Uchitel, J. Magee y J. Kramer, *Model-based Verification of Web Service Compositions*, En *Proceedings of the 18th IEEE International Conference on Automated Software Engineering*, Montreal - Canadá, (2003), pp 152-163.

[8] S. Narayanan y S.A. McIlraith, "Analysis and simulation of Web services", *Computer Networks*, Vol. **42**(5), pp. 675-693, (2003).

[9] S. Chun, y J. Offutt, *Generating Test Cases for XML-based Web Component Interactions Using Mutation Analysis*, En *Proceedings of the 12th IEEE International Symposium on Software Reliability Engineering*, Hong Kong - PRC, (2001), pp. 200-209.

[10] J. Offutt y W. Xu, "Generating Test Cases for Web Services Using Data Perturbation", *ACM SIGSOFT Software Engineering Notes*, Vol. **29**(5), pp. 1-10, (2004).

[11] H. Huang, W. Tsai, R. Paul y Y. Chen, *Automated Model Checking and Testing for Composite Web Services*, En *Proceedings of the 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, Seattle - USA, (2005), pp. 300-307.

[12] P. Ammann, P.E. Black y W. Majurski, *Using Model Checking to Generate Tests from Specifications*, En *Proceedings of the Second IEEE International Conference on Formal Engineering Methods*, Brisbane - Australia, (1998), pp 46-.

[13] J. Offutt, S. Liu, A. Abdurazik y P. Ammann, "Generating Test Data From State-based Specifications", *The Journal of Software Testing, Verification and Reliability*, Vol. **13**(1), pp. 25-53, (2003).

[14] J. García-Fanjul, J. Tuya y C. de la Riva, *Generating test cases specifications for BPEL compositions of web services using SPIN*, En *Proceedings of the International Workshop on Web Services - Modeling and Testing*, Palermo - Italia, (2006), pp. 83-94.

[15] J. García-Fanjul, C. de la Riva y J. Tuya, *Generation of Conformance Test Suites for Compositions of Web Services Using Model Checking*, En *Proceedings of "Testing: Academic & Industrial Conference - Practice and Research Techniques" (TAIC-PART)*, Windsor - UK (2006), pp. 127-130.