

Minimización de conjuntos de casos de prueba en la prueba de mutaciones de composiciones BPEL

Francisco Palomo Lozano, Antonia Estero Botaro e Inmaculada Medina Bulo *

Departamento de Ingeniería Informática, Universidad de Cádiz,
Avda. de la Universidad de Cádiz 10, 11519 Puerto Real, Spain
{francisco.palomo, antonia.estero, inmaculada.medina}@uca.es

Resumen Tanto en la aplicación de prueba de mutaciones a composiciones BPEL como en la realización de estudios experimentales sobre diversas métricas de calidad, surge la necesidad de minimizar conjuntos de casos de prueba manteniendo la máxima cobertura de mutación. La prueba de este tipo de software presenta algunas peculiaridades. Normalmente, las composiciones son relativamente pequeñas cuando se comparan con aplicaciones tradicionales, pues se encargan exclusivamente de la orquestación de los servicios, y no se dispone de un gran número de casos de prueba para ellas. No obstante, su ejecución puede resultar muy costosa, y debe realizarse para un número de mutantes que, normalmente, supera ampliamente al de casos de prueba. Se propone aquí como técnica de minimización una reducción a programación lineal entera y se evalúa su rendimiento para distintas composiciones.

Keywords: Minimización de conjuntos de casos de prueba, Programación lineal entera, Prueba de mutaciones, Composiciones BPEL, SBSE.

1. Introducción

La prueba de software supone una parte importante del coste de desarrollo que las estimaciones más conservadoras elevan hasta un 40%. Surge la necesidad de seleccionar cuidadosamente qué casos de prueba ejecutar, o en qué orden, para que se detecten errores potenciales al menor coste [9]. Existen dos situaciones en las que el coste de ejecución de las pruebas es muy alto, pero por motivos distintos. En la primera, la más común, se dispone de numerosos casos de prueba, probablemente heredados a lo largo de diferentes versiones. En la segunda, el conjunto de casos de prueba es relativamente pequeño, pero se emplea con un número elevado de características del sistema en pruebas (SUT).

Esta segunda situación es frecuente en el desarrollo de composiciones de servicios web. Las composiciones combinan servicios que se prueban por separado o cuya prueba no es responsabilidad de la organización que desarrolla o mantiene la composición. De hecho, puede que estén disponibles varios servicios y

* Parcialmente financiado por la red de excelencia TIN2015-71841-REDT (SEBASE-Net) y los proyectos TIN2014-60844-R (SAVANT) y TIN2015-65845-C3-3-R (DARDOS).

dinámicamente se seleccione entre ellos cuál ejecutar realmente en cada momento, dependiendo de una serie de condiciones. En conclusión, los servicios juegan para la composición el papel de cajas negras u oráculos, a los que se realiza una petición y devuelven un resultado. La labor del equipo de pruebas consiste en diseñar casos de prueba efectivos para la composición en sí. Estos pueden no ser numerosos, pero su coste de ejecución es generalmente muy alto.

No obstante, la eliminación de casos de prueba no puede abordarse efectivamente sin contar con métricas que permitan evaluar la efectividad de los casos de prueba seleccionados. Puesto que no existe un único criterio posible a la hora de fijar estas métricas e incluso pueden existir objetivos contrapuestos, es necesario especificar claramente el contexto. Presentamos aquí un enfoque para la minimización de conjuntos de casos de prueba basado en cobertura de mutaciones para composiciones WS-BPEL [7]. Para ello emplearemos técnicas exactas basadas en programación lineal entera (ILP).

2. Trabajos relacionados

Existe una vasta literatura en torno al empleo de técnicas metaheurísticas. No obstante, en aras de la brevedad, citaremos solo algunos trabajos relevantes que se han ocupado de las técnicas exactas de minimización de casos de prueba.

La mayoría de los autores que emplean técnicas exactas recurren a algún tipo de reducción de un problema de recubrimiento NP-difícil a otro problema objetivo, también NP-difícil, pero más general. Para estos problemas objetivo existen optimizadores muy eficientes que han sido desarrollados y mejorados durante décadas, en ocasiones con una fuerte inversión del sector privado.

Normalmente, el problema objetivo es ILP. Entre los trabajos en los que se aplica ILP a la minimización de casos de prueba pueden destacarse [4,2,10,8]. No obstante, un enfoque más reciente consiste en reducir la minimización a un problema SAT extendido, en lugar de a ILP, como en [1,6].

A diferencia del presente trabajo, la mayoría de autores supone un coste de ejecución idéntico para cada caso de prueba. Esto permite reducir el número de casos de prueba (por ejemplo, por subsunción) antes de abordar la minimización. Muchos usan la suite SIEMENS¹ donde la reducción tienen un gran impacto: tras aplicarla, el programa más complejo resultante es **replace**, con 215 casos de prueba y 208 características. Otro enfoque es el de [5], donde la métrica es la energía consumida durante el proceso de prueba; también emplea ILP.

3. Técnica de minimización

Aunque la técnica descrita a continuación es general y puede aplicarse a prácticamente cualquier SUT y marco de pruebas, se desarrollará específicamente para composiciones WS-BPEL empleando como marco de pruebas MuBPEL. Para una descripción de MuBPEL y de la terminología que sigue, véase [3].

¹ Disponible en <http://sir.unl.edu/portal>.

A partir de los registros producidos por MuBPEL para la composición se extrae la *matriz de ejecución*, E . Si se suministran los parámetros adecuados, los tiempos de ejecución de cada caso de prueba frente a cada mutante quedan registrados y puede obtenerse en paralelo la *matriz de costes* de ejecución, C . E y C son de dimensión $m \times n$, siendo $m = |M|$ y $n = |T|$.

En la matriz de ejecución, $e_{ij} = 1$ exactamente cuando m_i cubre a t_j o, equivalentemente, t_j mata a m_i . Si $e_{ij} = 2$ para algún j entonces m_i es un *mutante inválido*. En la matriz de costes, c_{ij} es el tiempo de ejecución del mutante m_i para el caso de prueba t_j . Las filas correspondientes a mutantes inválidos pueden eliminarse de E y C , por lo que puede suponerse que E es una matriz binaria y que los tiempos de prueba contenidos en C corresponden a ejecuciones válidas. También se elimina una fila si corresponde a un *mutante vivo*.

El objetivo es minimizar el número de casos de prueba en composiciones WS-BPEL empleando como métricas la *cobertura de mutaciones* y el *coste de ejecución*. En el primer escenario se debe preservar la máxima cobertura, mientras que en el segundo se debe además minimizar el coste global de ejecución de los casos de prueba seleccionados, esto es, el tiempo total de prueba. Ambos se reducen al siguiente ILP binario (BILP), donde las x_j son las variables de decisión asociadas a cada caso de prueba y, para el primer escenario, $c_{ij} = 1/m$:

$$\min \left\{ \sum_{1 \leq j \leq n} \left(x_j \sum_{1 \leq i \leq m} c_{ij} \right) \mid \forall i \in [1, m] \sum_{1 \leq j \leq n} e_{ij} \geq 1 \right\} \quad (1)$$

Para ello se ha programado un algoritmo en C++ que realiza la reducción al BILP correspondiente, lo resuelve mediante CPLEX, comprueba si este declara la solución producida como óptima e informa en caso contrario.

4. Resultados experimentales

Para los experimentos se dispone de cinco composiciones² WS-BPEL que mantiene el grupo UCASE. Para todas ellas se calcula la solución S con el mínimo número de casos de prueba y la cobertura de mutación máxima. Los tamaños y tiempos³ son los siguientes:

	A1	A2	A3	A4	A5
$ M $	96	890	213	508	3647
$ T $	8	34	19	37	95
$ S $	3	17	12	8	64
Tiempo (s)	0,01	0,03	0,02	0,02	2,80

El problema más complejo (A5) se ha resuelto también empleando como métrica el coste de ejecución. Se ha obtenido una solución óptima en tan solo 2,82s que permite mantener la máxima cobertura de mutación y minimiza el tiempo total de prueba, que se reduce de 185,80 h a 117,32 h.

² <https://neptuno.uca.es/redmine/projects/wsbpel-comp-repo/repository>.

³ Tiempos medidos en un Intel Core i5 5200U con 2,20 GHz y 8 GiB DDR3L

5. Conclusiones y trabajo futuro

La prueba de composiciones WS-BPEL puede presentar un elevado coste de ejecución por diversos factores: la necesidad de ejecutar cada caso de prueba en múltiples escenarios, la complejidad de algunos servicios o la propia naturaleza de la ejecución de las composiciones. La ejecución implica el despliegue y ejecución de una serie de servicios, servidores web y de aplicaciones, que han de realizarse para cada caso de prueba, lo que hace que al coste intrínseco a la ejecución de la prueba haya que sumar un coste fijo nada despreciable.

Aunque este trabajo se centra en las aplicaciones a la prueba de composiciones de servicios con WS-BPEL, la técnica propuesta es lo suficientemente general como para poder utilizarse con cualquier otro lenguaje de programación o marco de prueba, siempre que la ejecución de cada caso de prueba permita distinguir qué características del SUT se ejercitan. El rendimiento es bueno incluso cuando para la composición más compleja disponible en este estudio (A5) se mantiene la máxima cobertura de mutación y se minimiza no solo el número de casos de prueba, sino también el tiempo total de prueba. En el futuro se ampliarán los experimentos a un mayor número de composiciones.

Referencias

1. Arito, F., Chicano, F., Alba, E.: On the application of SAT solvers to the test suite minimization problem. *LNCS*, vol. 7515, pp. 45–59. Springer (2012)
2. Black, J., Melachrinoudis, E., Kaeli, D.: Bi-criteria models for all-uses test suite reduction. In: *Proc. 26th Int. Conf. on Soft. Engineering*. pp. 106–115 (2004)
3. Estero-Botaro, A., Palomo-Lozano, F., Medina-Bulo, I., Domínguez-Jiménez, J.J., García-Domínguez, A.: Quality metrics for mutation testing with applications to WS-BPEL compositions. *Softw. Test., Verif. Reliab.* 25(5–7), 536–571 (2015)
4. Fischer, K.: A test case selection method for the validation of software maintenance modifications. In: *Proceedings of International Computer Software and Applications Conference*. pp. 421–426. IEEE Computer Society Press (1977)
5. Li, D., Jin, Y., Sahin, C., Clause, J., Halfond, W.G.J.: Integrated energy-directed test suite optimization. In: *ISSTA 2014: Proceedings of the 2014 International Symposium on Software Testing and Analysis*. pp. 339–350. ACM (2014)
6. Lopez-Herrejon, R.E., Chicano, F., Ferrer, J., Egyed, A., Alba, E.: Multi-objective optimal test suite computation for software product line pairwise testing. In: *29th IEEE Int. Conf. on Software Maintenance*. pp. 404–407. IEEE (2013)
7. OASIS: Web Services Business Process Execution Language 2.0. <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html> (2007)
8. Wang, L., Wan, R., Wang, M., Li, M.: Generating small combinatorial test suite via lp. In: *WISA'09: Proc. of the 2009 Int. Symp. on Web Information Systems and Applications*. pp. 226–229 (2009)
9. Yoo, S., Harman, M.: Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability* 22(2), 67–120 (2012)
10. Zhong, H., Zhang, L., Mei, H.: An experimental study of four typical test suite reduction techniques. *Information and Software Technology* 50(6), 534–546 (2008)