Soporte Herramental para el Test Template Framework

Maximiliano Cristiá
CIFASIS-UNR y Flowgate Consulting
Rosario
cristia@cifasis-conicet.gov.ar

Resumen

Este proyecto trata sobre la automatización y extensión del Test Template Framework (TTF). El TTF es un método de testing basado en modelos (MBT) especialmente orientado a testing de unidad a partir de especificaciones Z. Aunque el TTF es un método sólido y fue ampliamente estudiado desde su primera publicación, la comunidad de MBT fue perdiendo interés en él. Nosotros creemos que esto se debió, al menos en parte, a la falta o dificultad aparente en dotarlo de un apoyo herramental. De hecho, algunos han sugerido que la generación de casos de prueba abstractos siguiendo el TTF es una actividad manual que requiere que los usuarios manipulen predicados complejos. La intención de este proyecto es mostrar que estas conclusiones son al menos dudosas, implementando una herramienta, llamada Fastest. Fastest no solo es capaz de producir automáticamente casos de prueba abstractos sino que además podría cubrir las necesidades de la comunidad Z en relación a herramientas de MBT.

Palabras clave: testing basado en modelos, test template framework, notación Z, Fastest

Contexto

El proyecto se desarrolla en el Centro Internacional Franco-Argentino de Ciencias de la Información y de Sistemas (CIFASIS) y en la empresa Flowgate Consulting. Hasta el momento el financiamiento ha provenido en su mayoría de Flowgate Consulting. Flowgate obtuvo parte del dinero para este proyecto a través de un ANR otorgado por FONTAR.

Por otro lado, durante los cuatro años que llevamos en este proyecto hemos interactuado con investigadores y profesionales de INVAP (Argentina), The Open University (Gran Bretaña), Instituto Nacional de Pesquisas Espaciais (INPE, Brasil), Instituto de Aeronáutica y Espaço (IAE, Brasil) y Altran Praxis (Gran Bretaña), con alguno de los cuales hemos realizado publicaciones conjuntas.

Introducción

El testing basado en modelos (MBT) es una técnica más o menos madura orientada a generar casos de prueba de un programa analizando su especificación formal [12, 9], como se muestra en la Figura 1. Estas técnicas

han sido desarrolladas y aplicadas a especificaciones escritas en diferentes notaciones formales como Z [11], máquinas de estados finitos y sus extensiones [8], B [10], especificaciones algebraicas [1], etc. La hipótesis fundamental detrás del MBT es que dado que un programa es correcto si verifica su especificación, entonces esta es una fuente excelente para obtener casos de prueba.

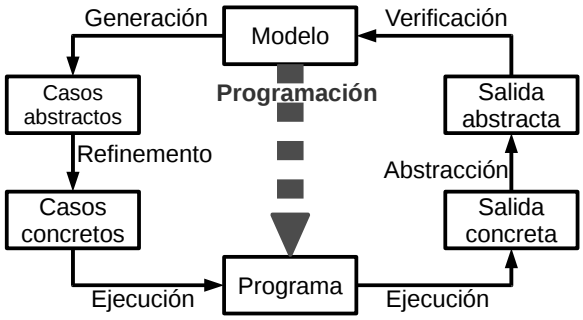

Figura 1: A general description of the MBT process.

Stocks y Carrington introdujeron un método de MBT basado en especificaciones Z llamado Test Template Framework (TTF) [11] que da una semántica particular al paso "Generación" de la Figura 1. El TTF apunta principalmente al testing de unidad. Nuestro grupo fue el primero en desarrollar una herramienta, llamada Fastest, que automatiza casi por completo el TTF [5, 3, 2]. Además podemos asegurar que es la primera implementación abiertamente disponible de un método de MBT para la comunidad Z. Una de las ventajas importantes del TTF para los usuarios de Z es que todos los conceptos importantes (clases de prueba, casos de prueba abstractos, etc.) se expresan en Z.

Según el TTF: (i) las especificaciones de tests se estructuran en *árboles de testing*, los cuales se generan aplicando *tácticas de testing*; (ii) los árboles de prueba deben ser podados para evitar especificaciones de tests insatisfacibles; y (iii) se deben derivar casos de prueba abstractos solo de las hojas de los árboles una vez podados. Todas estas actividades requieren complejas manipulaciones de predicados y esquemas Z que Fastest automatiza casi por completo.

Como parte de nuestro trabajo de investigación hemos aplicado Fastest a varios casos de estudio en colaboración con institutos de investigación o empresas. En particular trabajamos con investigadores del INPE en la aplicación de Fastest a problemas del área aeroespacial [6]. También hemos hecho lo propio con con el área de sistemas de INVAP e investigadores de IAE pero sin generar una publicación científica específica (se pueden ver los resultados en `http://www.flowgate.net/pdf/reports.tar.gz`). Estos resultados incluyen la formalización en Z de una porción no trivial del estándar aeroespacial ECSS-E-70-41A [7] a la cual se le aplicó Fastest lo que nos permitió descubrir limitaciones y fijar nuevos desafíos.

Líneas de Investigación y Desarrollo

Actualmente estamos trabajando en los siguientes sub-proyectos.

Traducción de casos de prueba a lenguaje natural

Los casos de prueba abstractos generados por Fastest son párrafos de texto Z, es decir texto formal. Si bien esta descripción de un caso de prueba es útil para automatizar el testing, requiere que los ingenieros involucrados en el proyecto sepan leer Z. En proyectos donde se requiere verificación independi-

ente esto puede ser un problema dado que los expertos externos pueden no saber Z. Al detectar este problema hemos comenzado a trabajar en un método semi-automático de traducción de los casos de prueba abstractos generados por Fastest a lenguaje natural [4]. Este sub-proyecto lo llevamos a cabo interactuando con investigadores del área de lingüística computacional de The Open University. El método propuesto entra dentro de la categoría de *generación de lenguaje natural basada en plantillas*.

Refinamiento de casos de prueba

Como dijimos más arriba, el TTF trata esencialmente sobre el paso "Generación" de la Figura 1; los autores originales no incluyeron en su trabajo el resto de los pasos del proceso de MBT. Nosotros hemos extendido el TTF completando el proceso de MBT. En particular hemos trabajado en la automatización del paso "Refinamiento" graficado en la Figura 1. En efecto, los casos de prueba generados por Fastest son abstractos en el sentido de que están escritos en lenguaje Z. En última instancia el propósito del MBT es testear un programa, por lo que los casos escritos en Z no son completamente útiles. Por esta razón es necesario refinarlos al nivel del programa. Esto significa traducirlos de Z al lenguaje de programación en el cual esté escrito el programa. Para tal fin hemos definido un *lenguaje de refinamiento de casos de prueba* (TCRL) mediante el cual se pueden escribir *reglas de refinamiento* independientes del lenguaje de programación. Además, implementamos un intérprete de ese lenguaje con una arquitectura tal que permite de manera sencilla refinar los casos a diferentes lenguajes de programación.

Aplicación y transferencia tecnológica

Nuestro objetivo de máxima es crear una tecnología que sea útil en la industria. Hasta el momento el testing funcional es una tarea prácticamente manual siendo automáticas solo sus etapas más sencillas (básicamente la ejecución de casos de prueba). En consecuencia gran parte de nuestro trabajo consiste en validar continuamente nuestros avances mediante la aplicación de Fastest a casos reales tomados de la industria.

En estos momentos estamos trabajando con la empresa Nemo Group (Argentina) para testear sus principales aplicaciones Java. Además, trabajamos en conjunto con ingenieros de Altran Praxis (Gran Bretaña) en la definición de nuevas funcionalidades de Fastest de manera tal de adecuar la herramienta a sus necesidades.

Formación de Recursos Humanos

Algunos de los miembros del equipo son docentes del Departamento de Ciencias de la Computación (DCC) de la Facultad de Ciencias Exactas, Ingeniería y Agrimensura (FCEIA) de la Universidad Nacional de Rosario (UNR); el resto son estudiantes o graduados de la Licenciatura en Ciencias de la Computación (LCC) del DCC.

Hasta el momento cuatro tesinas de grado se han desarrollado dentro del marco del proyecto. Cabe aclarar que estos estudiantes fueron becados por el grupo por lo que en algunos casos se les exigió un trabajo que normalmente excede la complejidad y longitud de una tesina de grado.

Referencias

[1] Gilles Bernot, Marie Claude Gaudel, and Bruno Marre. Software testing based on formal specifications: a theory and a tool. *Softw. Eng. J.*, 6(6):387–405, 1991.

[2] Maximiliano Cristiá, Pablo Albertengo, and Pablo Rodríguez Monetti. Fastest: a model-based testing tool for the Z notation. In Franco Mazzanti and Gianluca Trentani, editors, *PTD-SEFM*, pages 3–8. Consiglio Nazionale della Ricerche, Pisa, Italy, 2010.

[3] Maximiliano Cristiá, Pablo Albertengo, and Pablo Rodríguez Monetti. Pruning testing trees in the test template framework by detecting mathematical contradictions. In José Luis Fiadeiro and Stefania Gnesi, editors, *SEFM*, pages 268–277. IEEE Computer Society, 2010.

[4] Maximiliano Cristiá and Brian Plüss. Generating natural language descriptions of z test cases. In John D. Kelleher, Brian Mac Namee, Ielka van der Sluis, Anja Belz, Albert Gatt, and Alexander Koller, editors, *INLG*. The Association for Computer Linguistics, 2010.

[5] Maximiliano Cristiá and Pablo Rodríguez Monetti. Implementing and applying the Stocks-Carrington framework for model-based testing. In Karin Breitman and Ana Cavalcanti, editors, *ICFEM*, volume 5885 of *Lecture Notes in Computer Science*, pages 167–185. Springer, 2009.

[6] Maximiliano Cristiá, Valdivino Santiago, and N.L. Vijaykumar. On comparing and complementing two MBT approaches. In Fabián Vargas and Erika Cota, editors, *LATW*, pages 1–6. IEEE Computer Society, 2010.

[7] ECSS. Space Engineering – Ground Systems and Operations: Telemetry and Telecommand Packet Utilization. Technical Report ECSS-E-70-41A, European Space Agency, 2003.

[8] Wolfgang Grieskamp, Yuri Gurevich, Wolfram Schulte, and Margus Veanes. Generating finite state machines from abstract state machines. In *ISSTA '02: Proceedings of the 2002 ACM SIGSOFT international symposium on Software testing and analysis*, pages 112–122, New York, NY, USA, 2002. ACM.

[9] Robert M. Hierons, Kirill Bogdanov, Jonathan P. Bowen, Rance Cleaveland, John Derrick, Jeremy Dick, Marian Gheorghe, Mark Harman, Kalpesh Kapoor, Paul Krause, Gerald Lüttgen, Anthony J. H. Simons, Sergiy Vilkomir, Martin R. Woodward, and Hussein Zedan. Using formal specifications to support testing. *ACM Comput. Surv.*, 41(2):1–76, 2009.

[10] Bruno Legeard, Fabien Peureux, and Mark Utting. A Comparison of the BTT and TTF Test-Generation Methods. In *ZB '02: Proceedings of the 2nd International Conference of B and Z Users on Formal Specification and Development in Z and B*, pages 309–329, London, UK, 2002. Springer-Verlag.

[11] P. Stocks and D. Carrington. A Framework for Specification-Based Testing. *IEEE Transactions on Software Engineering*, 22(11):777–793, November 1996.

[12] Mark Utting and Bruno Legeard. *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.