TECNOTREND

ABS: UN LENGUAJE DE MODELADO EJECUTABLE Y ENFOCADO A SISTEMAS DISTRIBUIDOS Y ORIENTADOS A OBJETOS

Dra. Silvia Lizeth Tapia Tarifa

Nuestra articulista invitada, la Dra. Silvia Lizeth Tapia Tarifa es doctora por la Universidad de Oslo, en Noruega. Su principal área de investigación se enfoca en los métodos formales y la ingeniería de software. Actualmente, se encuentra desarrollando diversos proyectos de investigación relacionados con el ENVISAGE y el UpScale, integrándolo al lenguaje ABS, en la Universidad de Oslo.

Resumen

Este artículo da una introducción al lenguaje ABS [16, 13, 14], un lenguaje para el modelado de sistemas que tiene una semántica formal y una sintaxis similar a Java, este lenguaje adapta el modelo de concurrencia de lenguajes como Creol [17, 18] y ha sido desarrollado y utilizado por los proyectos europeos HATS [14] y ENVISAGE [12].

Año 2017 ISSN en trámite Universidad De La Salle Bajío A.C.
Reserva de Derechos al Uso Exclusivo No. 04 – 2016 – 113013104700 – 203

Palabras clave

Lenguaje ABS, desarrollo de software, sistemas.

Introducción

ABS apunta al modelado ejecutable de sistemas distribuidos y orientados a objetos. El modelado de sistemas es una técnica usada en el desarrollo de software, principalmente para entender sistemas ya existentes y que necesitan ser reemplazados, rediseñados o mejorados, o para diseñar nuevos sistemas.

Cuerpo

Un modelo es principalmente una representación simplificada del mundo real y, como tal, solo incluye algunos aspectos de interés que son relevantes para el problema en cuestión [7, 20].

Por lo tanto, los modelos contienen menos complejidad que las aplicaciones del mundo real, permitiendo entender más fácilmente esos aspectos de interés y abriendo la posibilidad de analizar propiedades específicas relacionadas a dichos aspectos. Al modelar sistemas concurrentes y distribuidos, la idea es entender y diseñar el comportamiento deseado derivado de la interacción de los componentes de estos sistemas.

Los modelos concurrentes abstraen muchos de los detalles que serían necesarios en la implementación real y se centran en la sincronización de procesos y en el comportamiento concurrente.

Muchos enfoques basados en modelos utilizan métodos formales [5, 15, 6, 3, 17, 2, 21], porque buscan aumentar la fiabilidad de los sistemas añadiendo rigor al proceso de diseño [8]. Los formalismos para el modelado y la comprensión de sistemas

Año 2017 ISSN en trámite Universidad De La Salle Bajío A.C.
Reserva de Derechos al Uso Exclusivo No. 04 – 2016 – 113013104700 – 203

concurrentes constituyen un campo de investigación activo en donde continuamente se proponen nuevos formalismos con diferentes características, tales como aumento en la expresividad; de tal modo, que un modelo concurrente sea más fácil de entender y demostrar que se comporta de acuerdo a las expectativas.

Cuando el modelado concurrente se combina con el enfoque orientado a objetos, como en el caso de ABS, es posible acercarse a la idea inicial de objetos propuestos por Dahl y Nygaard [9, 22], donde los objetos simulan entidades del mundo real que interactúan entre sí.

En este contexto, la orientación a objeto proporciona mecanismos de estructuración y flexibilidad que facilita la integración de cambios hechos a los objetos producidos durante el proceso de modelado, donde estos objetos son unidades independientes que interactúan por medio de llamadas a sus métodos [18, 17].

Con respecto al análisis, ABS combina características de objetos distribuidos con características del modelo de Actores, haciendo posible el uso de técnicas de razonamiento basado en la composición [11] que no son completamente automáticas, pero que pueden ser aplicadas en modelos más complejos y más cercanos a sistemas del mundo real.

A continuación resumiremos las principales características de ABS [13, 14]:

1. Tiene una sintaxis y semántica formal
2. Tiene una integración transparente entre concurrencia y orientación a objetos que se basa en grupos de objetos concurrentes o COGs, por sus siglas en inglés [16]
3. Permite que la comunicación entre objetos sea sincrónica y asincrónica [17, 10], parecida al modelo de Actores y parecido a como el lenguaje de

Año 2017 ISSN en trámite Universidad De La Salle Bajío A.C.
Reserva de Derechos al Uso Exclusivo No. 04 – 2016 – 113013104700 – 203

programación Erlang maneja sus procesos [4]

4. Ofrece una amplia variedad de alternativas de modelado complementarias usando un contexto concurrente y orientado a objetos.
5. Además integra tipos de datos algebraicos, programación funcional y programación imperativa
6. En comparación con lenguajes de programación orientados a objetos, permite abstraer la representación y manipulación de estructuras de datos
7. En comparación a los lenguajes gráficos como diagramas UML, es ejecutable y permite el modelado de flujo de control; de tal forma, que es sensible a datos de entrada, y
8. Soporta el modelado del despliegue mediante una separación de intereses entre costos y capacidades de recursos de ejecución (o máquinas virtuales) [19].

ABS cuenta con una página web en donde se pueden encontrar manuales, tutoriales y ejemplos [1]. Los simuladores de ABS se pueden descargar desde GitHub usando la siguiente dirección: `https://github.com/abstools/abstools`.

Adicionalmente, ABS cuenta con un entorno de pruebas en línea que permite acceder gratuitamente a todas las herramientas asociadas a este lenguaje: `http://ei.abs-models.org:8082/clients/web/`.

Referencias

[1] EU FP7 ENVISAGE. ABS Collaboratory: The ABS toolchain. http://abs-models.org/. Accedido: 2017-03-27.

[2] J.-R. Abrial. The B-book: Assigning Programs to Meanings. Cambridge University Press, New York, NY, USA, 1996.

Año 2017 ISSN en trámite Universidad De La Salle Bajío A.C.
Reserva de Derechos al Uso Exclusivo No. 04 – 2016 – 113013104700 – 203

[3] G. A. Agha. ACTORS: A Model of Concurrent Computations in Distributed Systems. The MIT Press, Cambridge, Mass., 1986.

[4] J. Armstrong. Programming Erlang: Software for a Concurrent World. Pragmatic Bookshelf, 2007.

[5] J. C. M. Baeten, T. Basten, and M. A. Reniers. Process Algebra: Equational Theories of Communicating Processes. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[6] C. Baier and J.-P. Katoen. Principles of Model Checking. The MIT Press, 2008.

[7] S. Beydeda, M. Book, and V. Gruhn. Model-Driven Software Development. Springer, 2005.

[8] H. Bowman and J. Derrick. Issues in formal methods (chapter 3). In H. Bowman and J. Derrick, editors, Formal Methods for Distributed Processing, A Survey of Object-oriented Approaches, pages 18–35. Cambridge University Press, 2001.

[9] O.-J. Dahl. The Roots of Object-orientation: The Simula Language. In M. Broy and E. Denert, editors, Software Pioneers, pages 78–90. Springer, 2002.

[10] F. S. de Boer, D. Clarke, and E. B. Johnsen. A complete guide to the future. In R. de Nicola, editor, Proc. 16th European Symposium on Programming (ESOP'07), volume 4421 of Lecture Notes in Computer Science, pages 316–330. Springer, Mar. 2007.

[11] C. C. Din, R. Bubel, and R. Hähnle. Key-ABS: A deductive verification tool for the concurrent modelling language abs. Automated Deduction - CADE-25: 25th International

Año 2017 ISSN en trámite Universidad De La Salle Bajío A.C.
Reserva de Derechos al Uso Exclusivo No. 04 – 2016 – 113013104700 – 203

Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings, 2015.

[12] EU FP7 ENVISAGE. Engineering Virtualised Services. http://www.envisage-project.eu/. Accedido: 2017-03-27.

[13] R. Hähnle. The abstract behavioral specification language: A tutorial introduction. In Formal Methods for Com- ponents and Objects, volume 7866 of Lecture Notes in Computer Science, pages 1–37. Springer, 2013.

[14] R. Hähnle, M. Helvensteijn, E. B. Johnsen, M. Lienhardt, D. Sangiorgi, I. Schaefer, and P. Y. H. Wong. Hats abstract behavioral specification: The architectural view. In B. Beckert, F. Damiani, F. Boer, and M. Bonsangue, editors, Formal Methods for Components and Objects, volume 7542 of Lecture Notes in Computer Science, pages 109–132. Springer, 2013.

[15] K. Jensen and L. M. Kristensen. Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer, 2009.

[16] E. B. Johnsen, R. Hähnle, J. Schäfer, R. Schlatte, and M. Steffen. ABS: A core language for abstract behavioral specification. In B. Aichernig, F. S. de Boer, and M. M. Bonsangue, editors, Proc. 9th International Symposium on Formal Methods for Components and Objects (FMCO 2010), volume 6957 of Lecture Notes in Computer Science, pages 142–164. Springer, 2011.

[17] E. B. Johnsen and O. Owe. An asynchronous communication model for distributed concurrent objects. Software and Systems Modeling, 6(1):35–58, Mar. 2007.

[18] E. B. Johnsen, O. Owe, and I. C. Yu. Creol: A type-safe object-oriented model for distributed concurrent systems. Theoretical Computer Science, 365(1–2):23–66, Nov. 2006.

[19] E. B. Johnsen, R. Schlatte, and S. L. Tapia Tarifa. Integrating deployment architectures and resource consumption in timed object-oriented models. Journal of Logical and Algebraic Methods in Programming, 84(1):67–91, 2015.

[20] J. Magee and J. Kramer. Concurrency: state models & Java programs. John Wiley & Sons, Inc., 1999.

[21] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science, 96:73– 155, 1992.

[22] O. Owe, S. Krogdahl, and T. Lyche. A biography of Ole-Johan Dahl. In O. Owe, S. Krogdahl, and T. Lyche, editors, From Object-Orientation to Formal Methods, volume 2635 of Lecture Notes in Computer Science, pages 1–7. Springer, 2004.

Año 2017 ISSN en trámite Universidad De La Salle Bajío A.C.
Reserva de Derechos al Uso Exclusivo No. 04 – 2016 – 113013104700 – 203