Integración de Modelos en UML y Especificaciones Formales: Transformaciones de OCL a RSL

Ana Funes, Arístides Dasso

Universidad Nacional de San Luis
Ejército de los Andes 950
5700 San Luis (Argentina)
Tel.: +54 2652 42 4027 int. 251
Fax: +54 2652 43 0059
{afunes, arisdas}@unsl.edu.ar

Resumen Continuando con nuestra investigación sobre la integración del Lenguaje Unificado de Modelado (UML) y el lenguaje de especificación del método formal RAISE (RSL), presentamos una de las líneas de trabajo en la cual nos encontramos investigando actualmente que consiste en la integración de OCL con RSL. Basándonos en nuestro trabajo previo, donde mostramos cómo obtener a partir de un diagrama de clases en UML una especificación inicial en RSL, nos encontramos construyendo un conjunto de reglas de transformación de restricciones de OCL a RSL.

Introducción

Los modelos construidos durante la etapa inicial del proceso de desarrollo de software son de gran importancia para el éxito futuro del proyecto ya que constituyen las bases sobre las que las etapas sucesivas se apoyan. Errores producidos en esta etapa tienen un gran impacto en los costes de todo el proyecto (Pressman, 2002).

La claridad y precisión de dichos modelos es fundamental no sólo por las razones antes mencionadas sino porque, como es sabido, durante esta etapa es cuando se espera el mayor grado de interacción entre el usuario final y el desarrollador.

Distintas notaciones gráficas, algunas más formales que otras, se han venido usando en la práctica para plasmar en modelos los requerimientos de los usuarios desde hace un largo tiempo (DeMarco, 1979; Chen, 1976; Harel, 1987; Jackson, 1982; Sutcliffe, 1988). Esto se debe fundamentalmente al hecho de que este tipo de notaciones son fácilmente transmitidas y captadas por el usuario final. En los últimos tiempos, los métodos orientados a objetos parecen ser los más usados en la práctica (Booch, 1991, 1994; Jacobson et al., 1992; Rumbaugh et al., 1991), y dentro de ellos aquellos que se basan en el uso del Lenguaje Unificado de Modelado (UML) como herramienta de modelado (Booch et al., 1999; Jacobson et al. 1999; Rumbaugh et al., 1999; OMG, 2003).

Por otro lado, sabemos que la construcción de modelos es una actividad compleja y creativa que tiende a contener inconsistencias y omisiones. Además, muchas de estas notaciones gráficas carecen de una semántica precisa lo que puede traer aparejado inconsistencias y ambigüedades en nuestros modelos.

Una práctica bien conocida para incrementar la confiabilidad del software es el uso de técnicas formales, sin embargo a pesar de que nos permiten escribir especificaciones libres de ambigüedades dificultan su validación con el usuario final.

Tomando los aspectos positivos de ambos enfoques es que se han llevado a cabo numerosas propuestas en el uso combinado de notaciones gráficas y técnicas formales. Una buena clasificación

de integraciones, incluyendo ejemplos, puede ser encontrada en el trabajo de Pons and Baum (Pons, 2000).

Trabajo Pasado, Presente y Futuro

En nuestro trabajo previo (Funes and George, 2003) hemos dado una propuesta de integración en el uso de los diagramas de clase de UML y RSL (The RAISE Language Group, 1992), el lenguaje de especificación del método RAISE (The RAISE Method Group, 1995).

En el trabajo mencionado mostramos cómo dicha integración es llevada a la práctica, dando la semántica de los diagramas de clase usando RSL como formalismo, aprovechando el hecho que RSL tiene su semántica definida formalmente (The RAISE Method Group, 1995). En consecuencia, por extensión, podremos afirmar que aquellos diagramas de clase en UML, que puedan ser representados en RSL, tendrán una semántica formal.

De esta manera, a partir de un modelo del sistema especificado por medio de un diagrama de clases en UML podemos derivar una especificación formal inicial en RSL. Esta especificación nos permite detectar inconsistencias en la especificación UML, además de permitirnos usarla como base para el desarrollo de las sucesivas fases del sistema de una manera formal, controlando en todo momento, gracias a las herramientas automáticas que posee el RSL, su consistencia.

Para soportar dicha transformación, hemos desarrollado una herramienta de software, llamada UML2RSL, que nos permite derivar en forma automática las especificaciones formales en RSL a partir de un diagrama UML. Una descripción de la herramienta puede ser encontrada en (Funes and George, 2003).

En el presente, continuando con la misma línea de investigación, hemos focalizado nuestro trabajo en la transformación del Lenguaje de Restricción de Objetos (OCL) (OMG, 2003; Warmer, J. et al., 1999) a RSL, con la idea de dar semántica formal a las expresiones en OCL empleando nuevamente la semántica formal del RSL.

OCL es una parte integral de UML que puede ser usado para especificar restricciones sobre los modelos construidos en UML. Por ejemplo, puede utilizarse para escribir invariantes sobre las clases y los tipos de un diagrama de clases, así como para dar las pre y post condiciones de las operaciones de una clase.

Nuestro objetivo es, entonces, encontrar un conjunto de reglas de transformación de OCL a RSL, para, de esa manera, poder integrar, por ejemplo, las invariantes en OCL que puedan incluirse en el diagrama de clases, como parte de la correspondiente especificación RSL, derivada de dicho diagrama, y permitir, así, un tratamiento conjunto de las construcciones del modelo con las restricciones en OCL.

El desarrollador podrá entonces mejorar sus modelos con restricciones sobre las clases y completar la definición de operaciones agregando anotaciones en OCL y luego obtener la correspondiente formalización en RSL, ya no sólo del diagrama sino también de las restricciones que el mismo tenga y que hayan sido incorporadas en OCL.

Por otro lado, además de la ya mencionada tarea de establecer las reglas de transformación de OCL a RSL, y una vez que las mismas sean consideradas que cumplen con una representación razonable y suficiente de OCL en RSL, planeamos incorporarlas como una funcionalidad adicional a la herramienta UML2RSL.

Referencias Bibliográficas

Booch, G. (1991). *Object-Oriented Design with Applications*. Benjamin/Cummings, Redwood City, CA.

Booch, G. (1994). *Object-Oriented Analysis and Design with Applications*. Benjamin/Cummings, Redwood City, CA, 2nd edition.

Booch, G., Rumbaugh, J., and Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Addison-Wesley.

Chen, P. (1976). The entity-relationship model - towards a unified view of data. *ACM Transactions on Database Systems 1*.

DeMarco, T. (1979). *Structured Analysis and System Specification*. Prentice-Hall.

Funes, A. and George, C. (2003). Chapter 8: "Formalizing UML class diagrams" in "UML and the Unified Process", ISBN 1931777446, Idea Group Publishing; April 1, 2003.

Harel, D. (1987). Statecharts: a visual formalism for complex systems. *Science of Computer Programming 8*, pages 231–274.

Jackson, M. (1982). *System Development*. Prentice-Hall.

Jacobson, I., Booch, G., and Rumbaugh, J. (1999). *The Unified Software Development Process*. Object Technology Series. Addison Wesley.

Jacobson, I. et al. (1992). *Object Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley,Reading, Ma.

OMG (2003). *OMG-Unified Modeling Language v1.5*, chapter UML Notation Guide. http://www.omg.org/technology/documents/formal/uml.htm

OMG (2003). *UML 2.0 OCL Specification*, http: //www.omg.org/technology/documents/formal/uml.htm.

Pons, C. and Baum, G. (2000). Formal Foundations of Object-Oriented Modeling Notations. In *Proceedings of 3rd IEEE International Conference on Formal Engineering Methods (ICFEM'00)*, September 04 - 07, 2000, York, England.

Pressman, R. (2002). *Ingeniería del Software, Un enfoque práctico*, Quinta Edición, Mc Graw Hill.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. (1991). *Object Oriented Modeling and Design*. Prentice Hall Inc., Englewood Cliffs.

Rumbaugh, J., Jacobson, I., and Booch, G. (1999). *The Unified Modeling Languge Reference Manual*. Object Technology Series. Addison Wesley.

Sutcliffe, A. (1988). *Jackson System Development*. Prentice-Hall.

The RAISE Language Group. (1992). *The RAISE Specification Language*. Prentice-Hall International (UK) Limited.

The RAISE Method Group.(1995). *The RAISE Development Method*. Prentice-Hall International (UK) Limited.

Warmer, J. and Kleppe, A. (1999). *The Object Constraint Language*. Object Technology Series. Addison Wesley.