Controles Semánticos para Modelamiento Orientado a Objetos

Susana A. Kahnert Pablo R. Fillottrani

Depto. Ciencias de la Computación
Universidad Nacional del Sur
Av. Alem 1253 – Bahía Blanca, Argentina
e-mail: {sak, prf }@cs.uns.edu.ar

1.- Introducción

En un sentido general, un modelo es "una abstracción de algo con el propósito de entenderlo antes de construirlo" [2]. Los modelos omiten detalles no esenciales, facilitando así el análisis de las propiedades de entidades demasiado complejas para ser entendidas directamente. Así como desde tiempos remotos las obras de ingeniería, arte o artesanía se construyen a partir de modelos, también para las distintas etapas del desarrollo de todo software resulta el Modelamiento una actividad crucial.

Es posible construir distintos modelos de un mismo sistema, cada uno enfatizando determinados aspectos que abstraen al sistema desde un punto de vista particular. Como cada uno de estos modelos enfoca la atención sobre determinados detalles mientras simplifica otros que no resulten importantes desde esa perspectiva, un sistema complejo podrá ser representado por una serie de pequeños modelos complementarios que facilitarán su entendimiento. En el proceso de desarrollo del software, un mismo modelo puede ser expresado a niveles diferentes de abstracción, según sea su propósito mostrar los requisitos y conocimiento del dominio, ayudar en el diseño del sistema, comunicar las decisiones de diseño, organizar un sistema complejo, documentar el sistema final, etc. Por consiguiente, la elección del modelo influirá profundamente en el enfoque del problema y en el aspecto de su solución.

Para que el modelo sea útil, es necesario formularlo en un lenguaje que pueda ser compartido por todas las personas involucradas en el desarrollo del sistema. Una propuesta en tal sentido es UML (Lenguaje Unificado de Modelamiento) [1,3]. UML es un lenguaje gráfico para visualizar, especificar, construir y documentar los elementos de un sistema de software, es decir para definir los modelos de ese sistema. E 1995 Booch, Jacobson y Rumbaugh combinaron sus propios lenguajes intentando conservar sus ventajas particulares, y crearon la primera versión de UML. En enero de 1997 UML se propuso como un estándar ante el Object Management Group (OMG). La propuesta fue modificada, y finalmente aprobada en noviembre del mismo año, como versión 1.1. Después de varias revisiones por el OMG, surgió la versión 1.3 en 1999.

La notación provista por UML para expresar un modelo incluye elementos gráficos y de texto. Adosada a estos elementos, hay una interpretación semántica que intenta capturar el significado del modelo. Las herramientas de soporte para el desarrollo de sistemas orientados a objetos en UML no sólo deberán proveer de un editor gráfico para la notación, sino también ayudar a asegurar la coherencia de sus aspectos semánticos. Esta última no es una tarea sencilla porque el significado podrá cambiar de acuerdo al propósito del modelo en un momento específico. Por ejemplo, a veces puede ser deseable verificar la consistencia de las relaciones entre las clases, pero después podría ser más importante asegurar la consistencia de tipos en cada invocación a métodos

.

En el grupo de Investigación y Desarrollo de Ingeniería de Software de la Universidad Nacional del Sur estamos trabajando actualmente en una herramienta, llamada HEMOO (Herramienta para Edición de Modelos Orientados a Objetos), para apoyar el desarrollo de modelos orientados a objetos usando UML. El objetivo es asistir al usuario en el manejo de los aspectos semánticos de un modelo, así como en los notacionales. En este sentido la herramienta proporcionará controles semánticos flexibles que puedan invocarse a pedido. Por invocación a pedido se entiende que la herramienta permitirá al usuario determinar cuáles de las propiedades definidas serán controladas, y en qué momento se efectuará ese control. La flexibilidad radica en la posibilidad que se le proporcionará al usuario de definir sus propios controles, que recibirán igual tratamiento que los controles predefinidos. Cabe esperar que este chequeo automático

simplifique significativamente la tarea de mantenimiento de la consistencia del modelo a través de todas las fases de su ciclo de vida, reduciendo al mismo tiempo la posibilidad de errores.

2. Controles semánticos en OOM

Dado que UML es un lenguaje de especificación, proporciona un vocabulario y un juego de reglas para combinarlo. Las unidades del vocabulario se dividen en tres categorías: los *elementos* que son abstracciones de las cosas que constituyen al modelo, las *relaciones* que conectan los diferentes elementos del modelo, y los *diagramas* que reúnen un conjunto de elementos relacionados. En particular, un diagrama es una representación gráfica de un conjunto de elementos, normalmente mostrado como un grafo conexo donde los nodos son los elementos y los arcos las relaciones. UML proporciona nueve tipos diferentes de diagramas, algunos de ellos para mostrar los aspectos estructurales, y otros para presentar la conducta del sistema,

Para construir estos diagramas, los diferentes elementos no se unen de una manera arbitraria. Además, dado que el mismo elemento puede aparecer en distintos diagramas, cada aparición del elemento en un diagrama debe ser consistente con las otras. Para contemplar estas situaciones, UML proporciona el concepto de *modelo bien formado*. Se dice que un modelo es bien formado cuando está correctamente construído y satisface todas las reglas y restricciones, predefinidas y específicas del modelo. En otros términos, un modelo bien formado es sintáctica y semánticamente correcto.

Las reglas pueden ser escogidas por el usuario para caracterizar en general un modelo bien formado, y la semántica particular para un modelo específico. El problema es que UML no caracteriza con precisión las reglas "predefinidas" que todos los modelos deben satisfacer. Es más, no hay ninguna manera clara de agregar reglas específicas a un modelo existente. Con respecto a las restricciones, UML permite al usuario escribirlas en cualquier lenguaje. Estas restricciones pueden adosarse a cualquier elemento en el modelo. Precisamente, la importancia de herramientas como HEMOO reside en ofrecer al usuario la posibilidad de adaptar el concepto de modelo bien formado según el contexto particular, y el momento particular en el proceso de desarrollo de software.

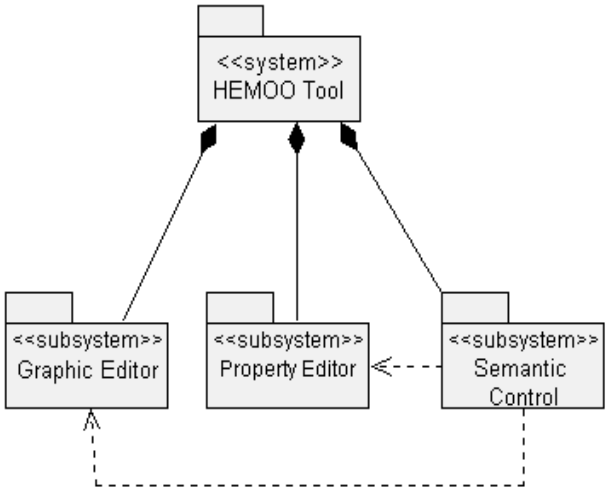
criterios	categorías
Generalidad	♦ Restricciones ♦ Reglas • Predefinidas • Específicas del modelo
Ambito	♦ elementos ♦ relaciones ♦ diagramas

Clasificación de los controles semánticos

3. la Herramienta HEMOO

El sistema está compuesto por tres subsistemas: Editor Gráfico, Editor de Propiedades, y Control Semántico. Ambos editores generan las representaciones internas para ser procesadas por el subsistema del Control Semántico.

El subsistema del Editor Gráfico tiene la responsabilidad de proveer al usuario las herramientas necesarias para definir un modelo en UML. Incluye facilidades para usar no sólo los elementos visuales, sino también los de texto, tales como restricciones, pseudocódigo y comentarios. Este subsistema genera una representación interna de los elementos en un modelo que usará el subsistema del Control Semántico.



Para que la consistencia de las restricciones, y también su compatibilidad con las reglas semánticas sea verificada por HEMOO, la herramienta proporciona una sintaxis precisa para describirlas. Este lenguaje se llama el HEMOO Lenguaje de Restricciones (HEMOOCL), y tiene asociado una semántica formal, de manera que pueda ser controlada su consistencia..

El subsistema del Editor de Propiedades es responsable de proporcionar las herramientas necesarias para introducir en el sistema las nuevas reglas definidas por el usuario. Esto no sólo incluye el nombre, y los posibles argumentos generales (tal como una clase o una relación), sino también su definición semántica. Para introducir la definición semántica de una regla, el subsistema permite al usuario especificar todas las propiedades existentes, "predefinidas" y definidas por el usuario, combinadas con elementos de programación lógica tales como conjunción, disyunción, negación por falla y predicados de segundo orden. La definición de propiedades así generada se usará en el Subsistema del Control Semántico.

El subsistema del Control Semántico es responsable de inducir al usuario a especificar las propiedades que quiere sean verificadas, e informar los resultados. Usa la representación del modelo y las propiedades generada por los otros dos subsistemas.

Actualmente estamos trabajando en una versión preliminar de la herramienta que puede manejar sólo Diagramas de Clase, que es el tipo más importante de diagrama en el modelamiento orientado a objetos; contiene clases, interfaces, paquetes, y las relaciones entre ellos.

La implementación del Editor Gráfico es en lenguaje Java, y la del Editor de Propiedades en Prolog. Hemos diseñado una barra de herramientas con elementos gráficos que le permitirán al usuario definir clases, interfaces y los distintos tipos de relaciones, y estamos implementando los medios para que el usuario pueda agregar atributos, operaciones, restricciones, y visualizar el diagrama de acuerdo al nivel de detalle que elija. En cuanto al Control Semántico, muchos de sus predicados ya están terminados y probados.

La posibilidad de extensiones futuras es amplia. Aunque los Diagramas de Clase son importantes, pueden agregarse otros tipos de diagramas para la definición de un modelo. En particular, como próximo paso planeamos extender el alcance de la herramienta al conjunto de todos los diagramas estructurales de UML

Referencias

[1] *The Unified Modeling Language User Guide* , Grady Booch, James Rumbaugh, Ivar Jacobson. Addison Wesley, 1999.

[2] *Object-Oriented Modeling and Design*, James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen. Prentice Hall, 1991.

[3] *The Unified Modeling Languae Reference Manual*, James Rumbaugh, Ivar Jacobson, Grady Booch. Addison Wesley, 1999.