Comprobación eficiente de restricciones de integridad en OCL

Jordi Cabot
Estudis d'Informàtica, Telecomunicació i Multimèdia
Universitat Oberta de Catalunta
jcabot@uoc.edu

Ernest Teniente
Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
teniente@lsi.upc.edu

Resumen

El proceso de comprobación de restricciones tiene como objetivo asegurar que el estado del sistema de información es consistente con las restricciones de integridad. La comprobación tiene que repetirse cada vez que el estado del sistema se modifica. La herramienta que aquí presentamos ayuda a realizar (automáticamente) esta comprobación de la forma más eficiente posible.

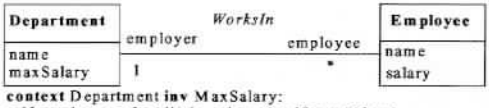
1. Introducción

Las restricciones de integridad (RIs) juegan un papel fundamental en la definición de los esquemas conceptuales (EC) ya que definen las condiciones que todo estado del sistema de información (SI) tiene que cumplir.

El estado del SI cambia debido a la ejecución de operaciones. El efecto de estas operaciones puede definirse indicando el conjunto de eventos estructurales que se aplican sobre los datos del SI durante su ejecución. Un evento estructural es un cambio elemental en la población de un tipo de entidad (o sea, clase) o de relación (asociación) como, por ejemplo, la inserción de un nuevo objeto, la modificación de un atributo, etc.

El SI tiene que garantizar que el nuevo estado resultante de la ejecución de una operación es consistente con las RIs definidas en el EC. Este proceso se conoce como *comprobación de restricciones de integridad* y debe ser tan eficiente como sea posible. Normalmente la eficiencia se consigue mediante el análisis de los eventos definidos en la operación. Este análisis permite asegurar la consistencia del nuevo estado sin tener que reevaluar completamente todas las RIs con el objetivo es considerar el mínimo número posible de objetos durante su comprobación.

Por ejemplo, la restricción *MaxSalary* del EC de la Figura 1 (que obliga a que el salario de un empleado sea menor que el salario máximo definido en su departamento) puede violarse cuando se ejecuta la operación *AddEmployeeToDept(e:Employee, d:Department)* encargada de crear una nueva relación entre el empleado *e* y el departamento *d* (ya que el salario de *e* puede ser mayor que el salario máximo en *d*). No obstante, no es necesario evaluar todos los departamentos para comprobar *MaxSalary* después de ejecutar esta operación. Basta con considerar *d*. Además, entre todos los empleados trabajando en *d*, sólo *e* puede violar la restricción.




```
context Department inv MaxSalary:
self.employee->forAll(e| e.salary<=self.maxSalary)
```

Figura 1. EC de ejemplo

Nuestra herramienta ofrece diversas funcionalidades para mejorar la eficiencia de la comprobación de restricciones de integridad en ECs especificados en UML/OCL ya que es capaz de 1- Proporcionar información valiosa al diseñador (como qué tipos de eventos pueden inducir la violación de una RI) y 2 – Generar automáticamente un EC extendido que, cuando se implementa en cualquier plataforma tecnológica, comprueba todas las RIs de forma eficiente.

Gracias a que la herramienta trabaja a nivel conceptual es independiente de tecnología. Por lo tanto, puede integrarse en cualquier herramienta *MDA-compliant* para mejorar la eficiencia del proceso de comprobación sin importar la plataforma tecnológica donde finalmente se implemente el EC. Es importante resaltar que, actualmente, el soporte para RIs en herramientas MDA es bastante limitado [2].

Actas de las XII Jornadas de Ingeniería del Software y Bases de Datos, JISBD'2007, pp.391-392 ISBN: 978-84-9732-595-0

2. Descripción de la herramienta

La herramienta se compone de un conjunto de clases Java que reutilizan las librerías del *Dresden OCL Toolkit* y de *NetBeans MDR* para transformar el fichero XMI de entrada (representando el EC) y la definición textual de las RIs en una representación interna basada en los metamodelos de UML y OCL. Esta representación interna es la utilizada para implementar las diferentes funcionalidades de la herramienta, que se describen a continuación. En [1] puede encontrarse la herramienta así cómo información detallada de cada funcionalidad.

2.1. Eventos que pueden violar una RI

La herramienta puede determinar qué eventos pueden violar una RI. Como se ve en la figura 2, sólo la modificación de los atributos *maxSalary* y *salary* y la inserción de una nueva relación (i.e. link) en *WorksIn* pueden violar *MaxSalary*. Por lo tanto, podemos mejorar la eficiencia de la comprobación si obviamos evaluar *MaxSalary* después de ejecutar operaciones que no incluyan ninguno de estos eventos.

2.2. Expresiones incremental para evaluar RIs

Para minimizar el número de instancias que intervienen en la comprobación de una RI *r* después de un evento *ev*, la herramienta computa una expresión incremental *exp* que puede usarse en lugar de *r* durante la verificación de la consistencia del SI después de ejecutar *ev*. La Figura 2 muestra la expresión para comprobar *MaxSalary* después del evento "modificación del salario de un empleado". Según la expresión, en este caso basta con evaluar la relación entre el empleado modificado (representado por la variable *self*) y su departamento.

2.3. Generación de un EC eficiente

El diseñador es responsable de sacar partido de la información proporcionada por las funcionalidades anteriores durante la implementación del EC. No obstante, nuestra herramienta es también capaz de modificar automáticamente el EC inicial para crear un EC' que en ejecutarse (o implementarse directamente) sea capaz de comprobar todas las RIs eficientemente. Básicamente, EC' se crea extendiendo EC con nuevos tipos de entidad (para guardar la información sobre los eventos ejecutados des de la última comprobación) y redefiniendo las RIs de forma sólo se comprueben para las instancias afectadas por esos eventos.

Referencias

[1] Cabot, J., Teniente, E.: A Tool for the Incremental Evaluation of OCL Constraints: www.lsi.upc.edu/~jcabot/research/IncrementalOCL

[2] Cabot, J., Teniente, E.: Constraint Support in MDA tools: a Survey. ECMDA'06, LNCS, 4066, 256-267

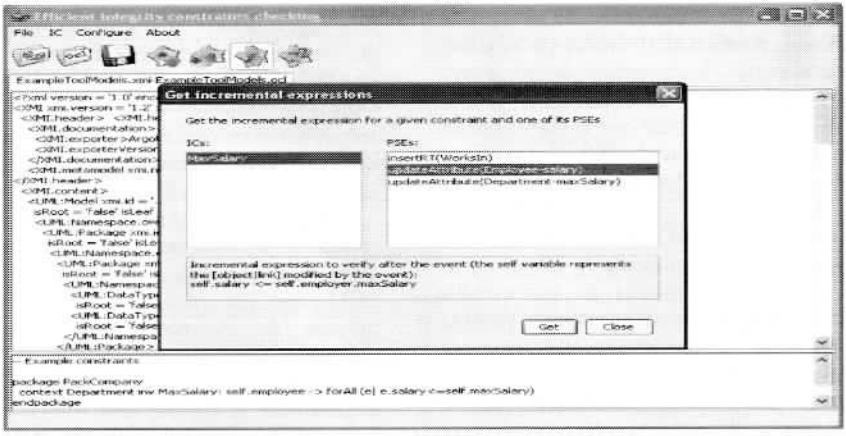


Figura 2. Screenshot de la herramienta