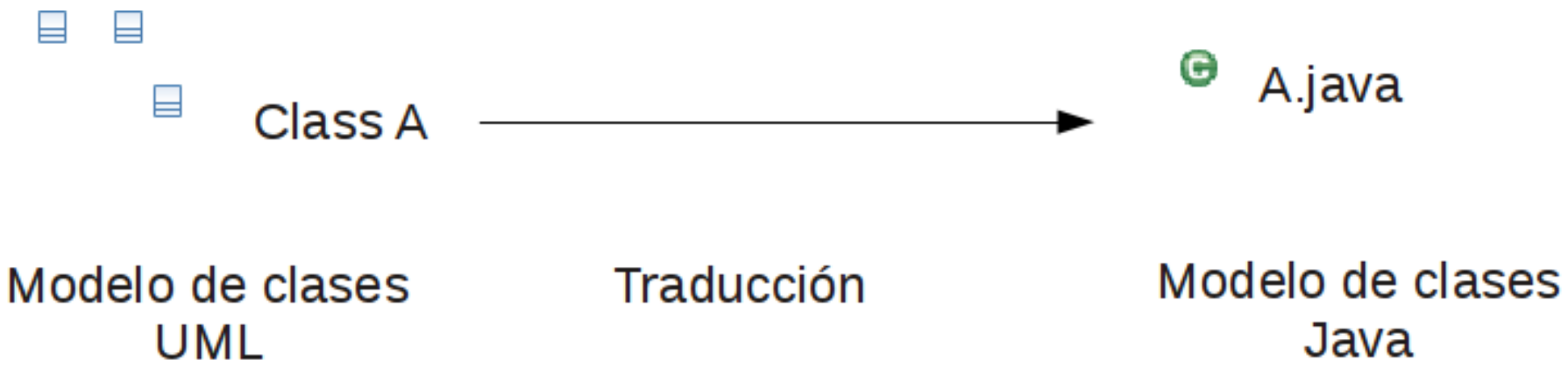
Desarrollo de una herramienta para derivación automática de especificaciones OCL a JML

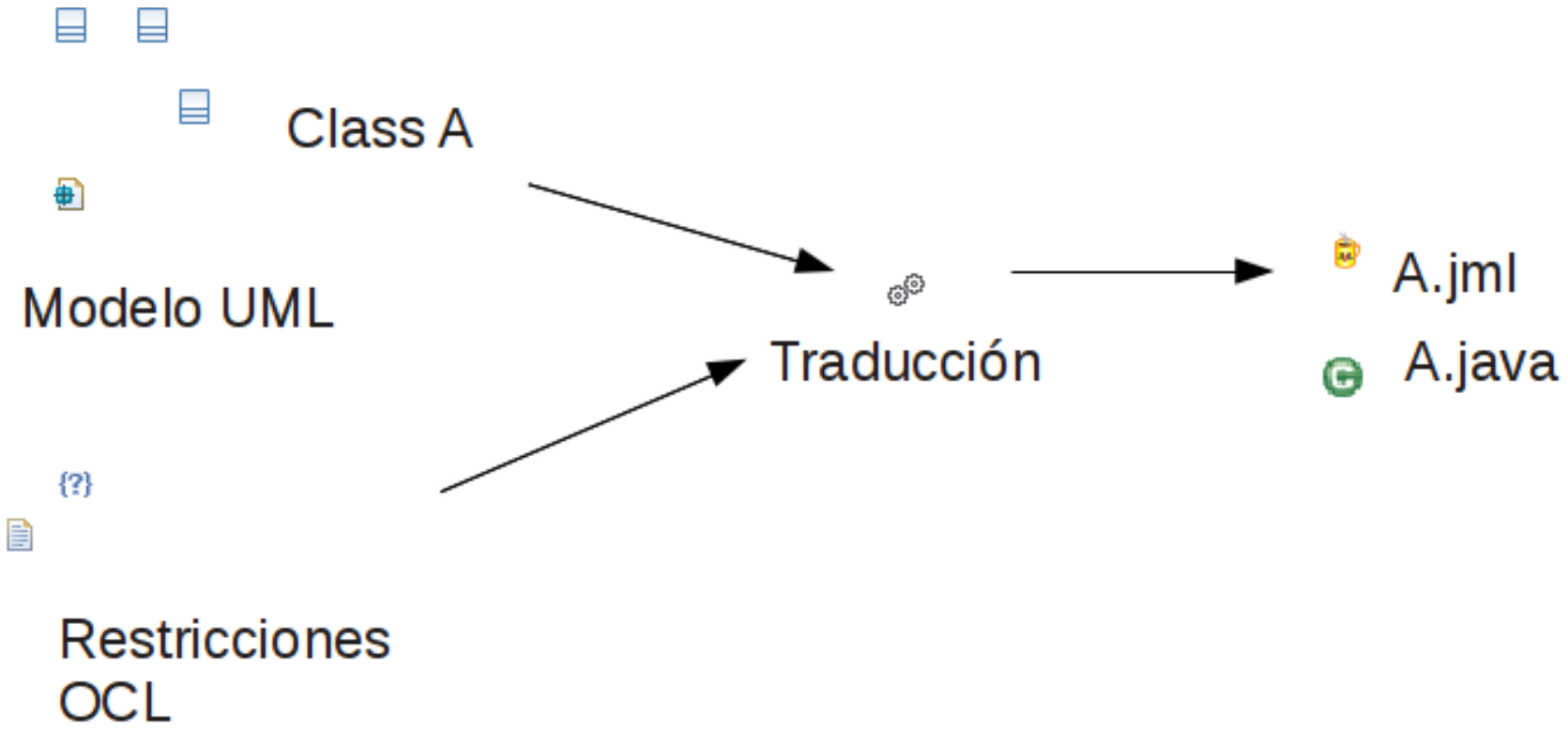
María José Dias Molina Diego Matías Dodero Mena

29 de Noviembre de 2011

Introducción



Introducción



Objetivos

- Investigar métodos formales para la especificación de programas.
- Comparar los lenguajes de especificación existentes.
- Investigar herramientas de verificación de programas.
- Integrar JML con OCL dentro del marco del desarrollo orientado a modelos.
- Investigar la factibilidad de la traducción automática de OCL a JML.

OCL (Object Constraint Language)

- Lenguaje de especificación.
- Refina modelos UML.
- No tiene efectos laterales.
- Es un lenguaje tipado.

OCL (Object Constraint Language)

Las restricciones que tenemos en OCL son:

- Las invariantes se escriben con la cláusula *inv*.
- Las definiciones se escriben con la cláusula *def*.
- Las precondiciones se escriben con la cláusula *pre*.
- Las postcondiciones se escriben con la cláusula *post*.

JML (Java Modeling Language)

- Lenguaje de especificación para Java diseñado en 1999.
- Utiliza el estilo de Diseño por contrato, usando pre y post condiciones.
- No tiene efectos laterales.
- Puede ser usado por cualquier programador Java.

JML (Java Modeling Language)

El Diseño por contrato en JML se utiliza de la siguiente forma:

- Las precondiciones se escriben con la cláusula *requires*.
- Las postcondiciones se escriben con la cláusula *ensures*.
- Las invariantes se escriben con la cláusula *invariant*.

Ventajas y desventajas de OCL

- Ventajas
  1. Tiene un nivel más alto de abstracción.
  2. Es independiente del lenguaje.
  3. Está estandarizado por el OMG.
- Desventajas
  1. OCL no puede verificarse fácilmente en tiempo de ejecución.

Ventajas y desventajas de JML

- Ventajas
  1. Similitud al lenguaje Java.
  2. Provee conceptos a nivel de implementación.
  3. Existen herramientas de verificación que utilizan anotaciones JML.
- Desventajas
  1. Está relacionado al lenguaje Java.
  2. No está estandarizado.

Diferencias semánticas entre OCL y JML

- Estado previo a la ejecución
- Frame conditions
- Cuantificadores
- Aritmética
- Igualdad e identidad
- Tipo de dato arreglo
- Excepciones
- Niveles de visibilidad

Traducción

- Tipos simples
- Operadores y expresiones
- Pseudovariables
- Cuantificadores
- Let
- Contratos
- Colecciones

Construcciones no traducidas

- Iteradores *collect*, *iterate*, *sortedBy*, *any*.
- Método *allInstances*
- Tuplas
- Expresión de invocación de mensaje en una postcondición.

Herramientas

- **Verificadores en tiempo de ejecución**
  - jmlrac
  - jmlunit
- **Verificadores estáticos de código**
  - ESC/Java2
  - Sireum/Kiasan
  - OpenJML
- **Buscadores de modelos**
  - JMLForge
- **Verificadores de programas**
  - KeY
  - Loop
  - Jack

ESC/Java2

- Herramienta de verificación.
- Realiza un análisis estático.
- Encuentra posibles errores en tiempo de ejecución.

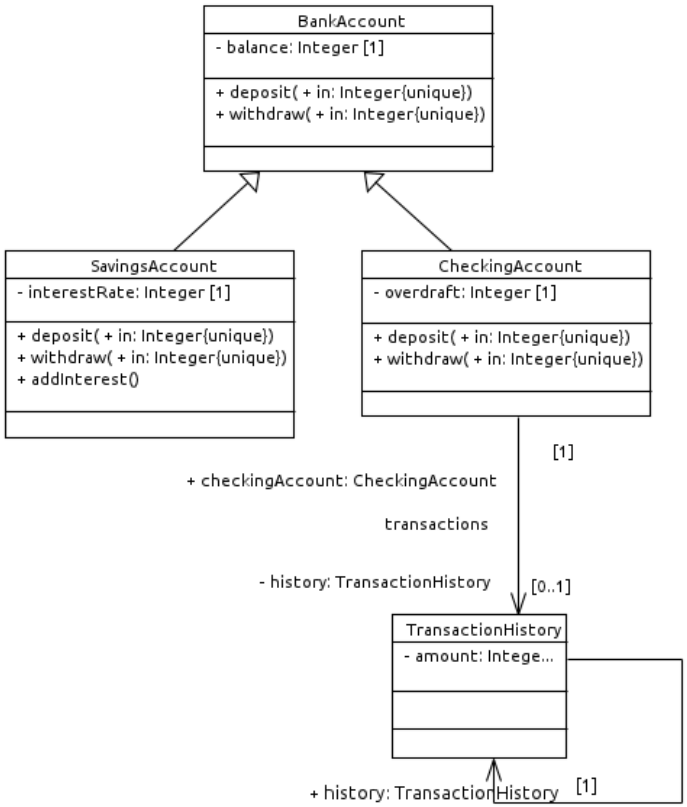
ESC/Java2 - Ventajas

- Automática, no requiere intervención humana.
- Se obtiene feedback sin especificaciones.
- Detecta la mayoría de los errores comunes.
- Fuerza al programador a escribir los contratos de los métodos.

ESC/Java2 - Desventajas

- No es completa ni consistente.
- Se requiere cierta experiencia para interpretar los resultados.
- Sólo procesa código fuente Java hasta la versión 1.4.

Diagrama de ejemplo



Trabajos relacionados

- A library-based approach to translating OCL constraints to JML assertions for runtime checking
  - OCL no puede verificarse en tiempo de ejecución.
  - Separar código Java de especificaciones JML.
  - Link: http://digitalcommons.utep.edu/cs_techrep/71/
- Projeto Compilador OCL - JML Geração de Código JML
  - Compilador de OCL a JML
  - Link: http://code.google.com/p/ocl-jm-compiler/

Trabajos relacionados. Continuación

- Herramienta para automatizar la transformación UML/OCL a Object-Z (Tesina).
  - Link: http://catalogo.info.unlp.edu.ar/meran/opac-detail.pl?id1=2132
- Definición formal de la semántica de UML-OCL a través de su traducción a Object-Z.
  - Presenta traducción de UML con restricciones OCL a expresiones Object-Z.
  - Link: http://www.lifia.info.unlp.edu.ar/papers/2003/Becker2003.pdf

Trabajos relacionados. Continuación

- Towards Verifying Java Realizations Of OCL-Constrained Design Models Using JML.
  - Utilizar JML como lenguaje de aserciones.
  - Verifica los requerimientos OCL a través de las especificaciones JML.
  - Utilizar un demostrador de teoremas.
  - Link: http://eprints.brighton.ac.uk/2218/1/SEA02.pdf
- Strategies for Translating UML/OCL Design Models to JAVA/JML Designs
  - Traducción automática de modelo UML/OCL a JAVA/JML.
  - Utilizar herramientas JML.
  - Link: http://dspace.lib.fcu.edu.tw/jspui/browse?type=author&value=Hamie Ali&value_lang=zh_TW

Trabajos relacionados. Continuación

- Translating the Object Constraint Language into the Java Modeling Language
  - Mapeo de expresiones OCL a restricciones JML.
  - Similitud al lenguaje Java.
  - Existencia de herramienta JML.
  - Link: http://dl.acm.org/citation.cfm?id=968206

Conclusiones

- ¿Qué logramos?:
  - Derivación automática de OCL a JML.
  - Separar la generación de las anotaciones JML de la generación del código Java.
  - Ejecutar la verificación sobre el código generado.
- ¿Qué problemas tuvimos?:
  - Inconvenientes tecnológicos:
    - Falta de madurez de los plugins y herramientas.
    - Incompatibilidad de versiones.
    - No existe estandarización de JML.
- Ventajas:
  - Simple mantenimiento de las especificaciones.
  - La traducción automática es menos propensa a errores.

Trabajos futuros

- Investigar la posibilidad de utilizar una transformación M2M para realizar la traducción.
- Implementación de las operaciones de colecciones de OCL restantes.
- Integrar un editor de modelos a la herramienta.
- Integración de las herramientas de verificación al plugin desarrollado.

Muchas Gracias.

¿Preguntas?