Impacto de reglas de refactorización en diagramas UML con restricciones OCL

Trabajo de Grado
Licenciatura en Informática – Plan 90

Alumno: Santiago Scolari
Directora: Claudia Pons

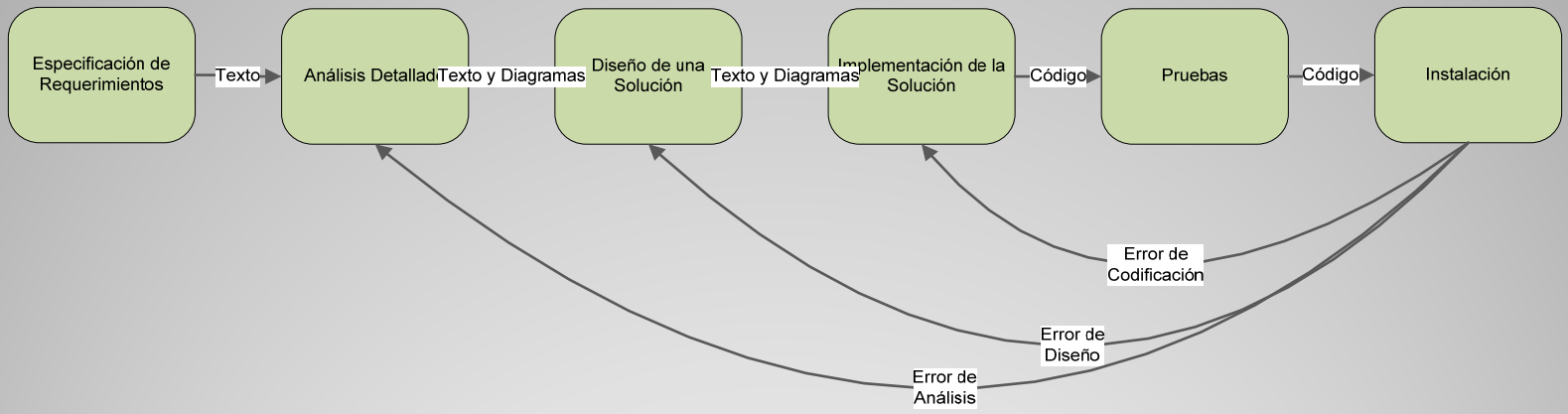
Facultad de Informática – UNLP – Febrero de 2010

Índice

- Introducción
- Motivación
- Objetivos
- MDA
- UML
- OCL
- Refinamientos
- Problema del código OCL en refactorizaciones
- Nuevo catálogo de reglas de refactorización
- ePlatero
- Modulo refactorizaciones OCL en ePlatero
- Ejemplos de uso
- Conclusiones

Introducción

- Desarrollo de software tradicional



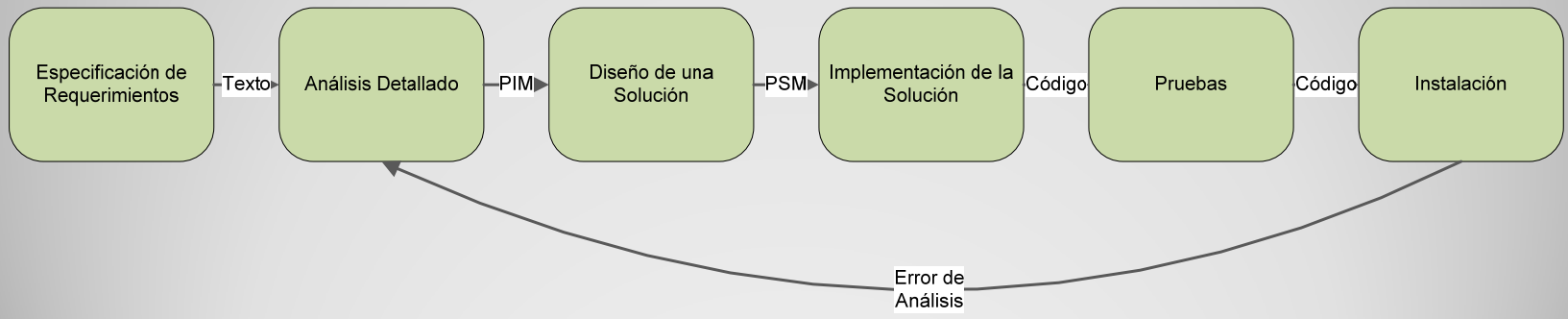
- Problemas
  - Problema de la productividad
  - Problema de la portabilidad
  - Problema de la interoperabilidad
  - Problema del mantenimiento y la documentación.

Introducción

- MDA (Model Driven Architecture)
  - Es una implementación de MDD (Model Driven Development)
- Desarrollo de software dirigido por modelos
  - Los modelos en MDA son escritos en UML.
  - Estos son transformados y van evolucionando.
  - Mediante sucesivas transformaciones se llega al nivel de detalle requerido para generar código ejecutable.

Introducción

- Proceso de desarrollo de software con MDA
  - Solución a los problemas del desarrollo tradicional
  - Modificaciones al proceso de desarrollo



Motivación

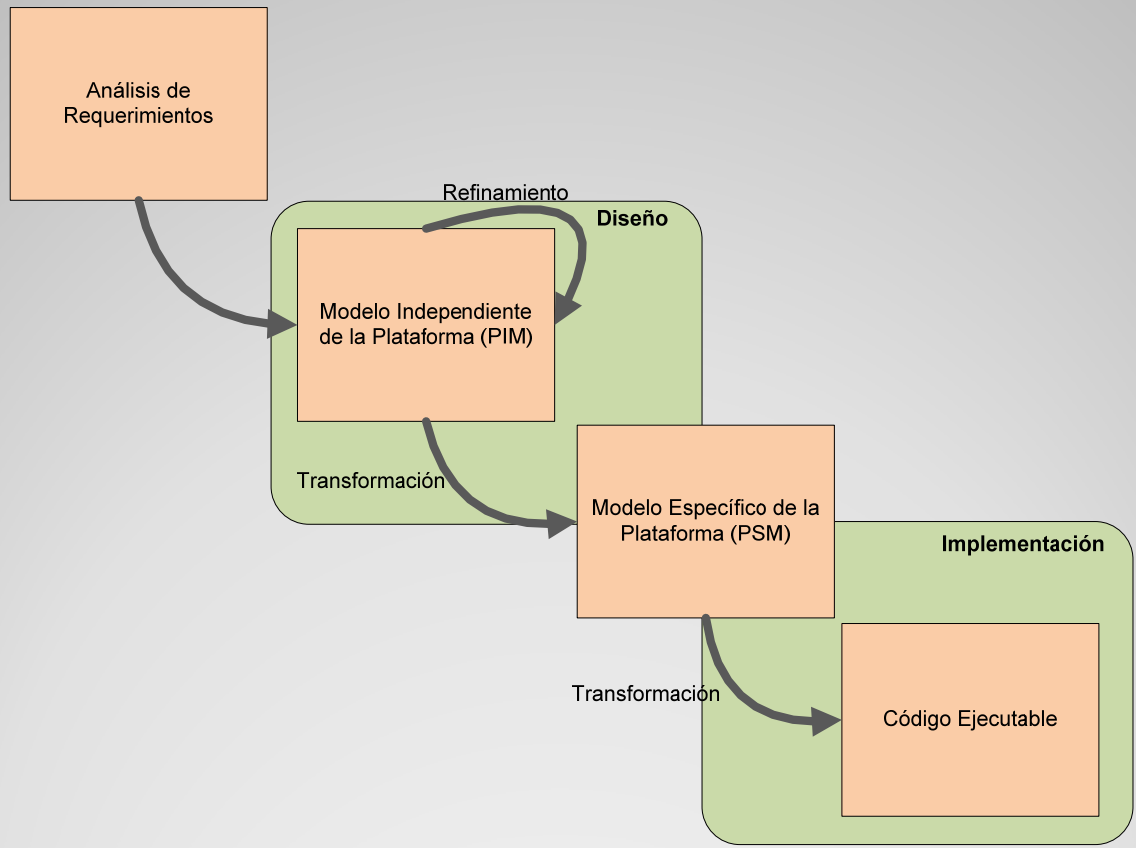
- Los sistemas están en constante evolución
  - Mucho esfuerzo invertido en la evolución y mantenimiento
- MDA permite desarrollar software a partir de modelos.
  - Los modelos permiten un fácil mantenimiento y evolución.
- Los modelos en MDA son escritos en UML, y complementados con OCL.
- Los estudios sobre refactorizaciones
  - Se basan en diagramas UML
  - No tienen en cuenta las restricciones OCL asociadas.

Objetivos

- Hacer un aporte a la propuesta planteada por MDA.
- Estudiar las reglas de transformación de diagramas UML con restricciones OCL
  - Definir un conjunto de reglas para tener en cuenta las restricciones OCL
- Contribuir a las herramientas de soporte de MDA.
  - Implementar en ePlatero la solución encontrada.

MDA

- El proceso MDA



UML

- UML es un estándar de facto en la industria del software.
- Es un lenguaje de modelado y documentación.
- Es sólo una notación, no define un proceso de desarrollo.
- Diferentes procesos de desarrollo lo utilizan.
- UML es el lenguaje elegido por MDA para la construcción de los modelos.
- Los modelos y las transformaciones de los mismos están definidos en UML.

OCL

- Lenguaje formal para la expresión de restricciones.
- Es parte de UML.
- A pesar de ser un lenguaje formal, es fácil de leer y escribir.
- Permite expresar restricciones del sistema que no se pueden expresar gráficamente en diagramas UML.

OCL

- Restricciones
  - Invariantes
  - Definiciones
  - Precondiciones
  - Postcondiciones
- Expresiones de valor inicial
- Expresiones de valor derivado
- Expresiones de consulta

Modelos

- En el desarrollo con MDA, la principal tarea es la construcción de modelos.
- Un modelo es una abstracción de un sistema.
- Es esencial producir modelos de calidad.
- En diferentes proyectos se utilizan los lenguajes de modelado de diferentes maneras.
- Niveles de madurez de modelado
  - MML 0: Sin especificación
  - MML 1: Textual
  - MML 2: Texto con modelos
  - MML 3: Modelos con texto
  - MML 4: Modelos precisos
  - MML 5: Solo modelos

Modelos

- MML 4
  - En este nivel los modelos tienen relación directa con el código fuente
    - Es posible realizar generar automáticamente gran parte del código
  - Los cambios se realizan sobre el modelo, y a continuación se regenera el código.
  - Código y modelos se mantienen actualizados.
  - Los modelos son más difíciles de generar
- MDA requiere modelos de nivel 4
- La mejor opción para construir modelos MML 4 es UML con OCL.

Refinamientos

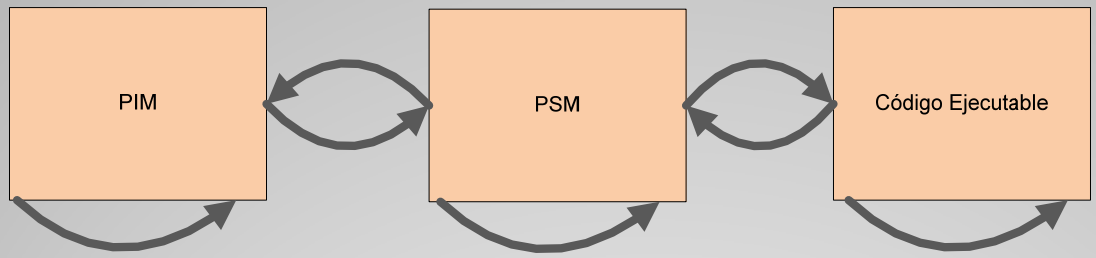
- MDA define los refinamientos como el proceso de transformar modelos.
- Tipos de transformaciones
  - PIM a PIM
    - Agregan o quitan información del modelo.
    - Pasaje del análisis al diseño.
    - Transformación de un modelo en otro mas detallado.
    - No son automáticas.
    - Caso de estudio de esta tesis.

Refinamientos

- Otras transformaciones
  - PIM a PSM
    - Se realizan cuando los PIM son lo suficientemente completos
    - Desde un PIM se genera una implementación del sistema en una tecnología en particular.
    - Se utilizan reglas de transformación
    - Altamente automatizable
  - PSM a PIM
    - Se realiza sobre un PSM de una aplicación existente.
    - Ingeniería inversa.
    - Muy difíciles de realizar.
  - PSM a PSM
    - Muchas veces una sola transformación de PIM a PSM no es suficiente.
    - Optimizaciones del código fuente generado.

Refinamientos

- Transformación de modelos



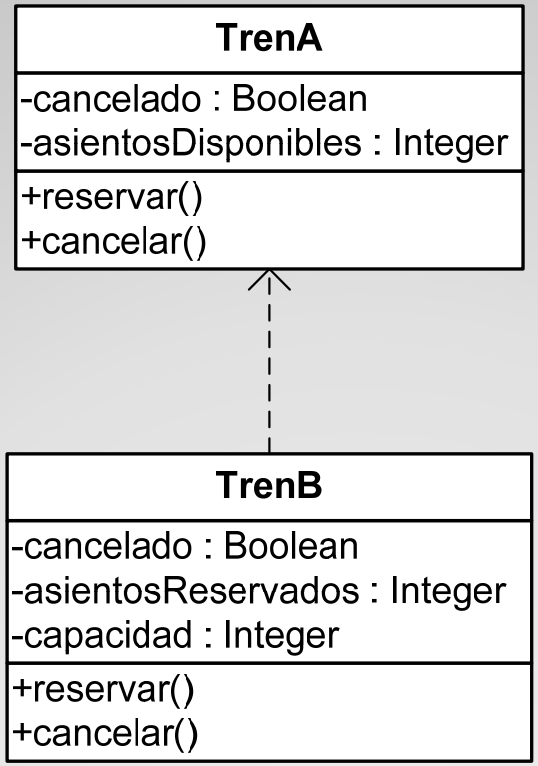
- Causas para la refactorización de modelos
  - Requerimientos
  - Patrones de diseño
  - Decisiones técnicas
  - Requerimientos de calidad
- Diferentes versiones de un modelo son creadas con distinto nivel de abstracción.
- Es imprescindible asegurar que los modelos conserven consistencia al ser refinados.

Refinamientos

- Caso de estudio
  - Transformaciones de PIM a PIM
    - Realizadas de UML a UML
    - Es una rama reciente de investigación
    - Se tienen en cuenta cambios a nivel de
      - Clases
      - Atributos
      - Métodos.
      - No se tienen en cuenta las restricciones OCL.
  - Es necesario que el código OCL evolucione conjuntamente con el UML para garantizar la coherencia de los diagramas

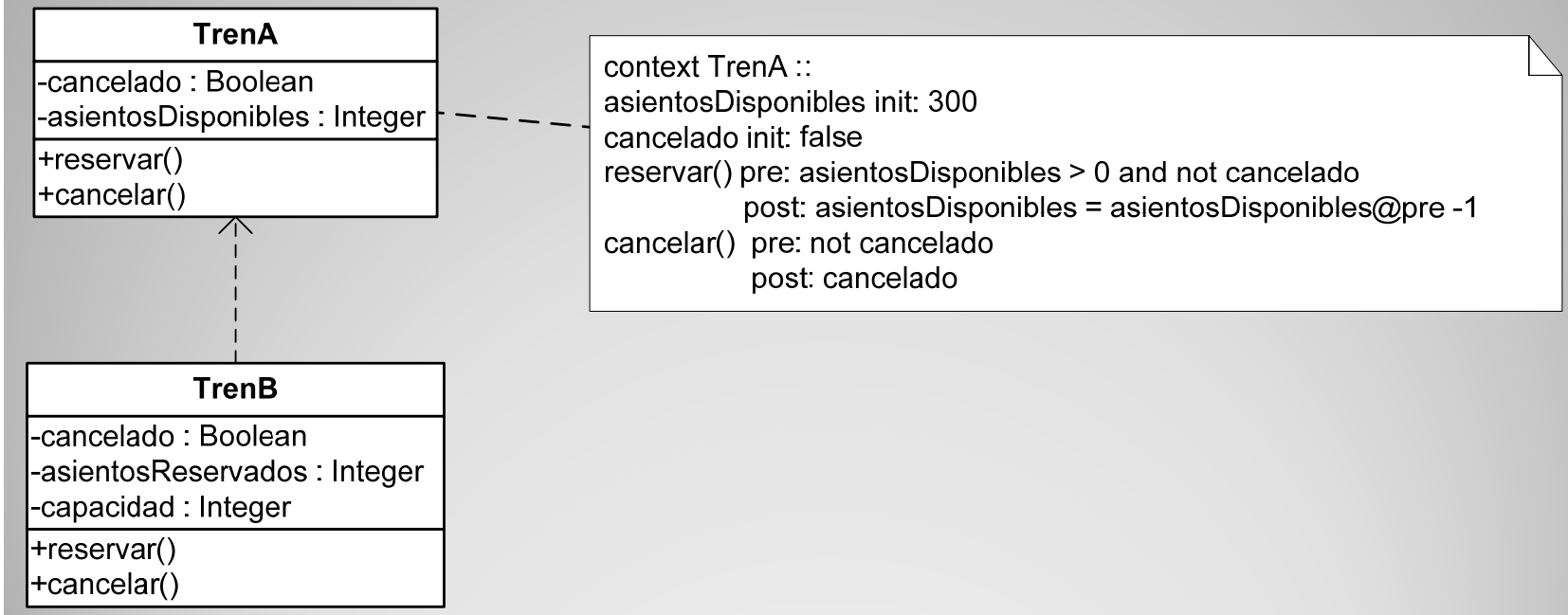
Problema del código OCL

- Ejemplo: Diagrama UML sin restricciones OCL



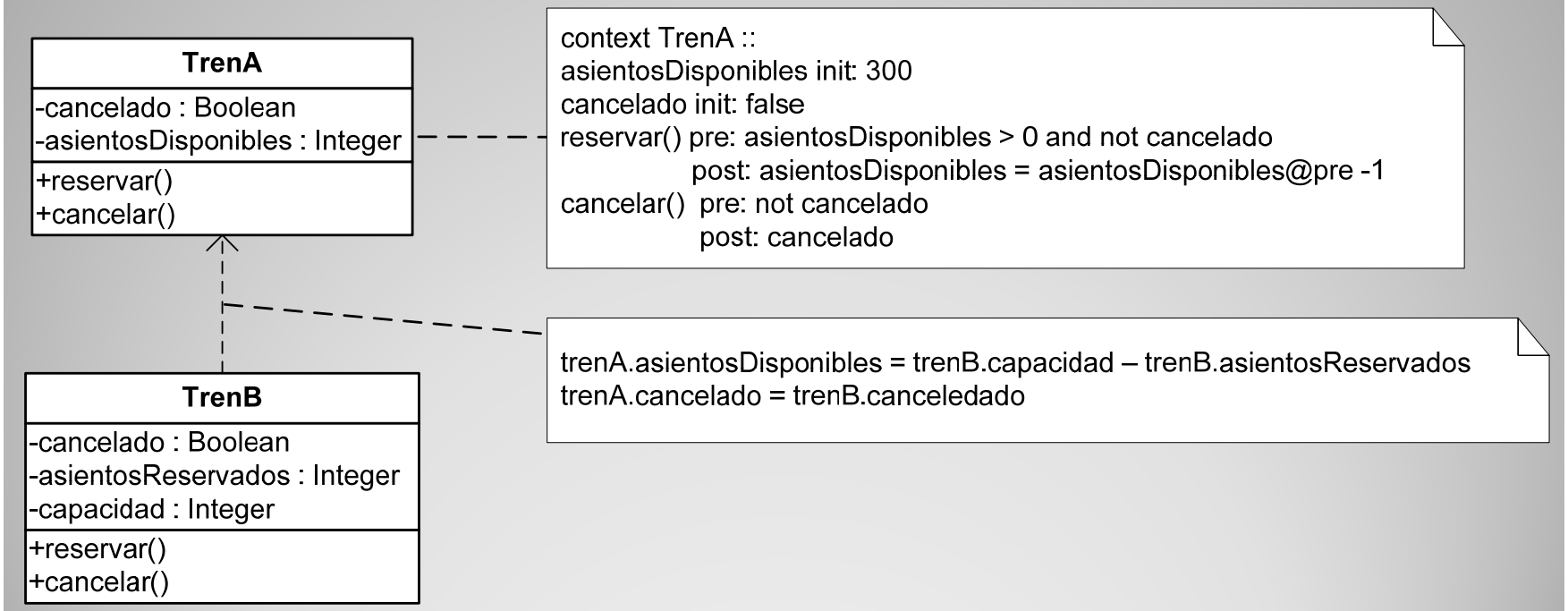
Problema del código OCL

- Versión mejorada con el agregado de OCL



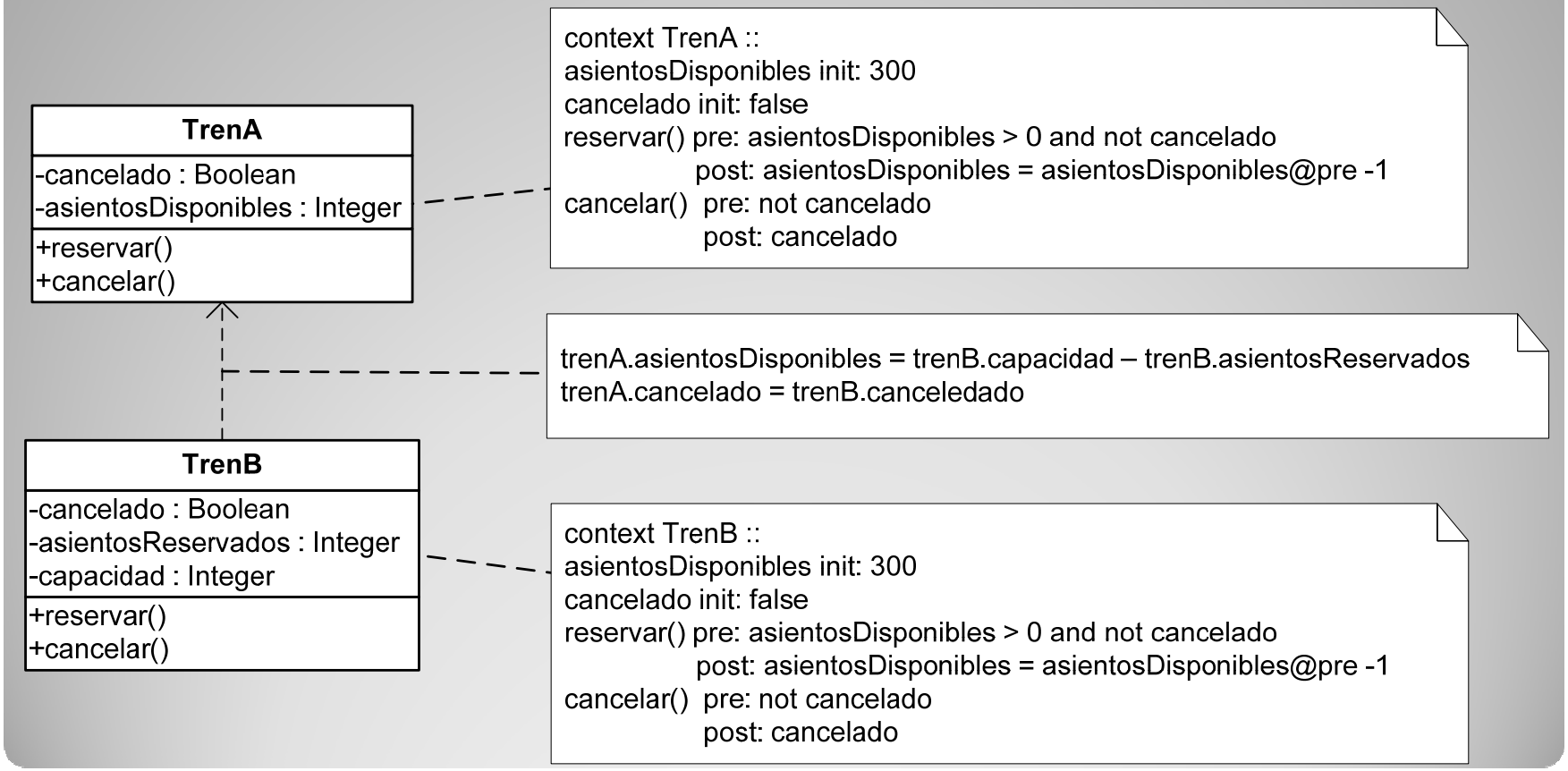
Problema del código OCL

- Agregando OCL en el mapeo del refinamiento



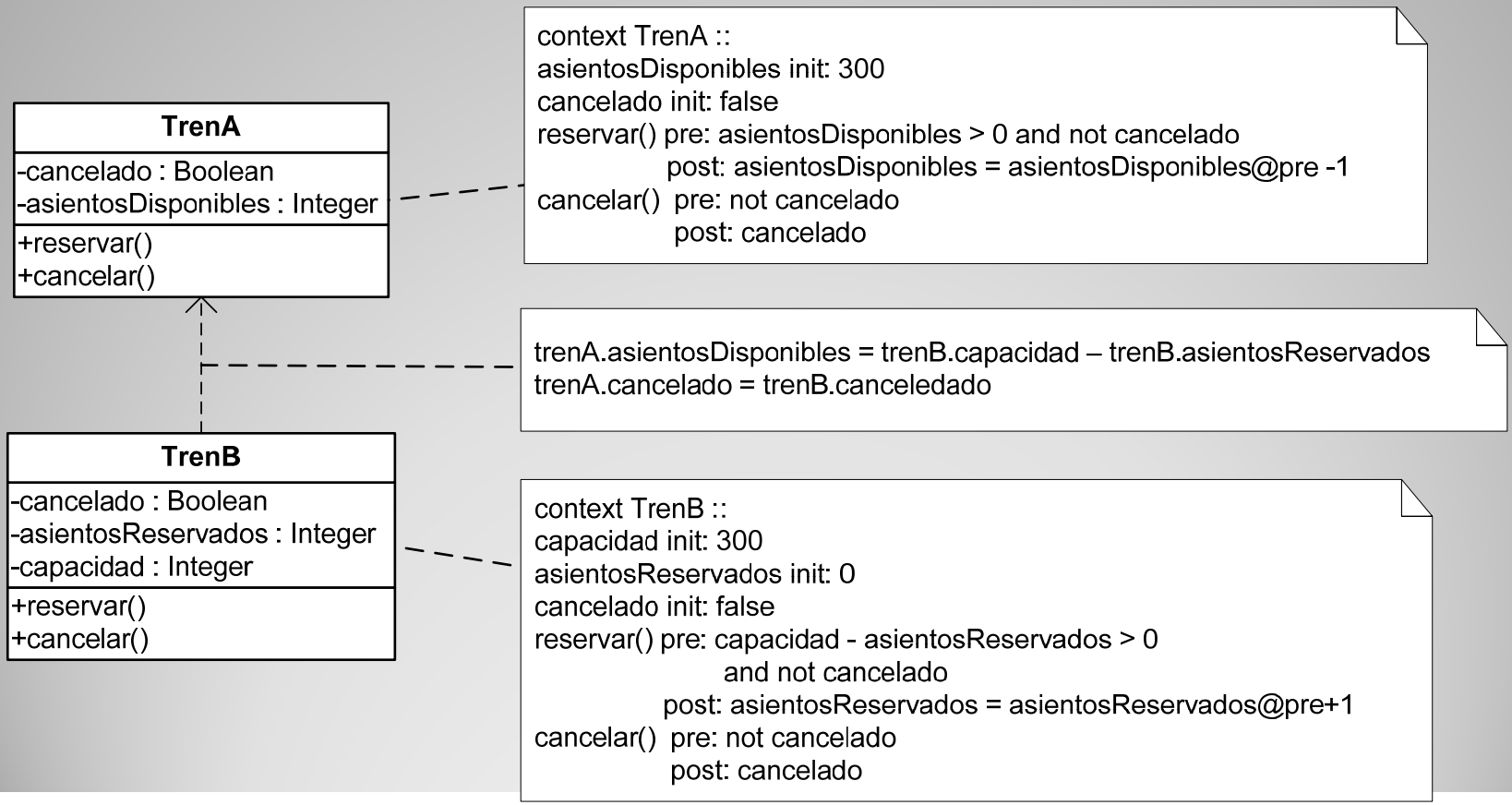
Problema del código OCL

- OCL de TrenB??



Problema del código OCL

- Versión corregida

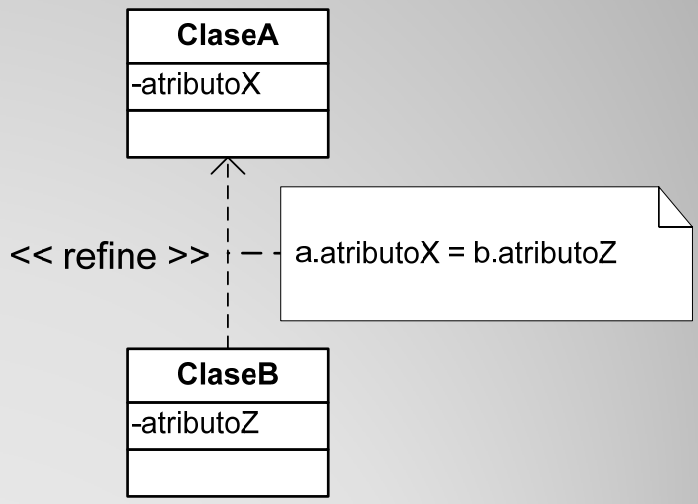


Problema del código OCL

- Solución al problema
  - Transformaciones de código OCL
  - Catalogo de reglas
    - A partir del mapeo de la refactorización, se inferirán las reglas de transformación de OCL.
    - Al aplicar estas reglas en el código OCL original, dará como resultado el código OCL de la clase refactorizada.
  - Implementar esta solución en una herramienta de soporte de MDA.

Catálogo de reglas

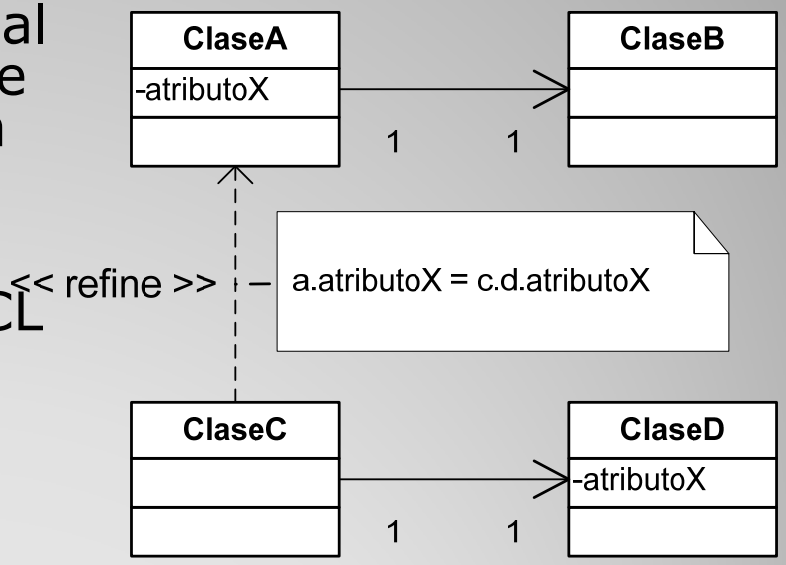
- Renombrar atributo
  - Cambia el nombre de un atributo
  - El atributo aparece en la clase refactorizada con un nombre distinto al original.



- Cambios OCL:
  - En las restricciones OCL de la nueva clase, el nombre del atributo es cambiado en cada ocurrencia .

Catálogo de reglas

- Mover atributo
  - Un atributo de la clase original pasa a ser parte de otra clase después de la refactorización

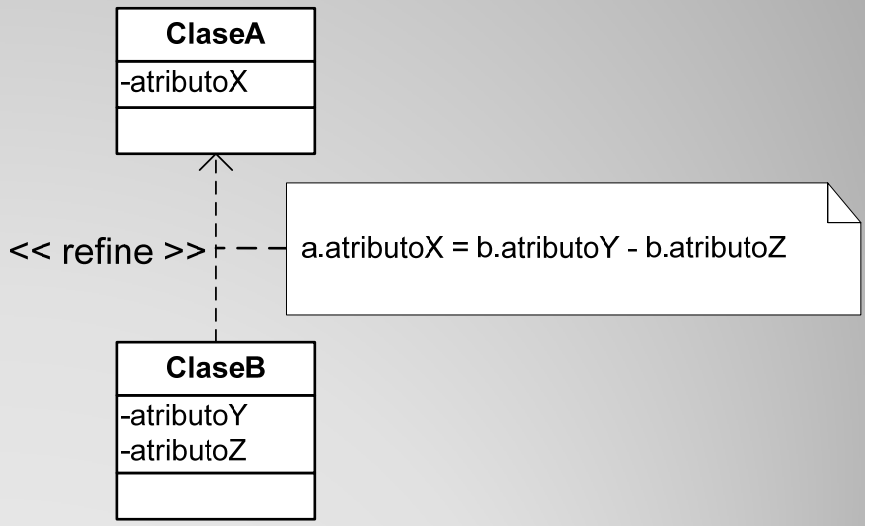


- Cambios OCL
  - Actualiza las restricciones OCL en todas las localizaciones donde se utiliza la variable movida
  - Navegación hacia adelante:
    - Las expresiones de la forma ***a.atributoX*** tienen que ser rescritas de la forma ***b.c.atributoX***

Catálogo de reglas

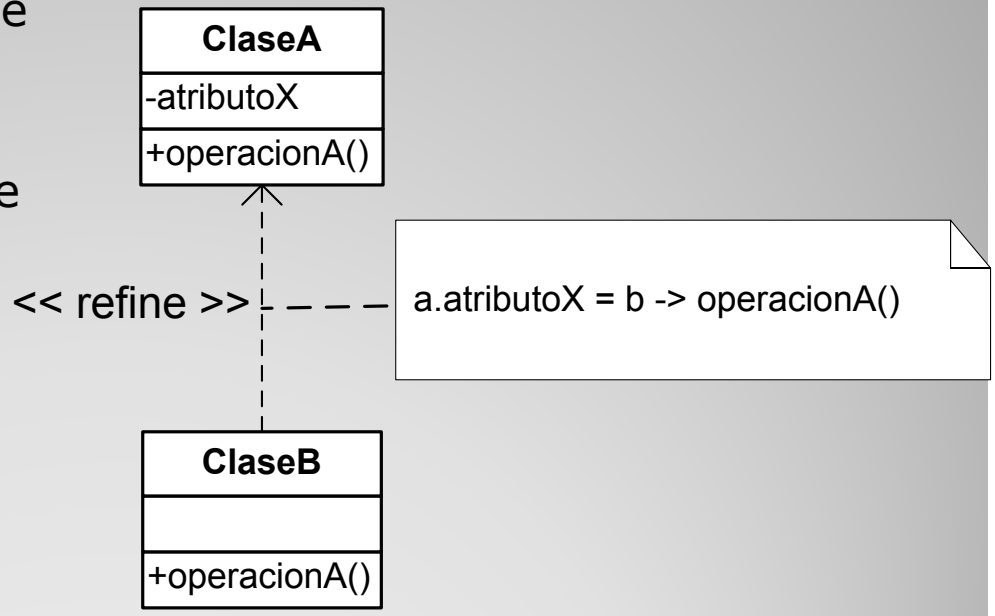
- Refactorizar atributos
  - Transforma un atributo en dos.

- Cambios OCL
  - Se crean los nuevos atributos ***atributoY*** y ***atributoZ***.
  - El atributo ***atributoX*** se elimina.
  - Se mantiene la relación entre ambos luego de la refactorización
  - El valor por defecto debe ser despejado para obtener los valores de los nuevos atributos
  - Donde aparezca el atributo refactorizado, este será reemplazado por la nueva relación



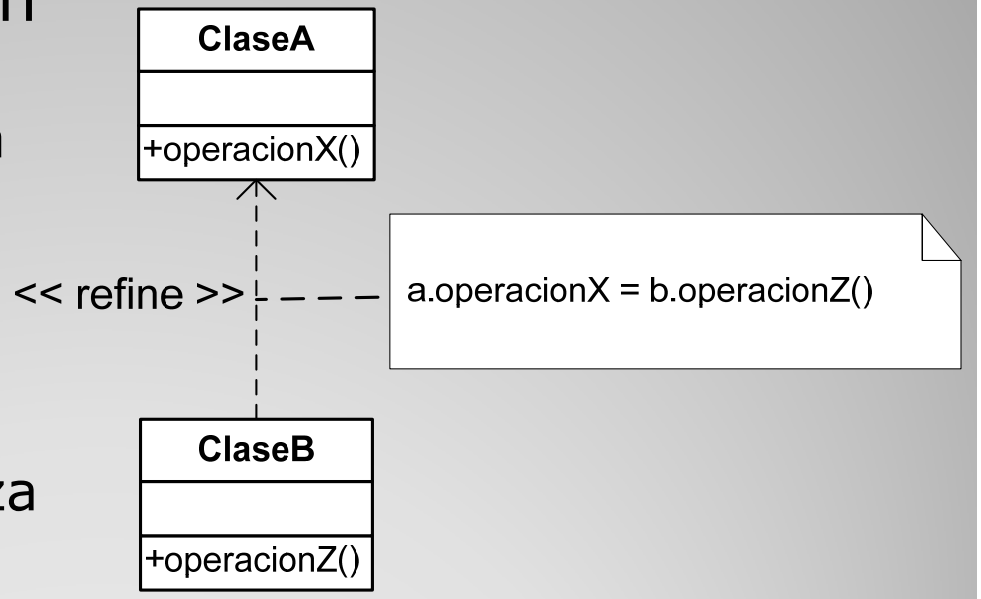
Catálogo de reglas

- Transformar atributo en valor calculado
  - Un atributo se elimina y se reemplaza por un valor calculado.
  - Para el calculo se utilizan atributos y operaciones de la refactorización.
- Cambios OCL
  - Restricciones de inicialización, se eliminan
  - Expresiones donde aparezca la variable, se reemplaza por el valor calculado



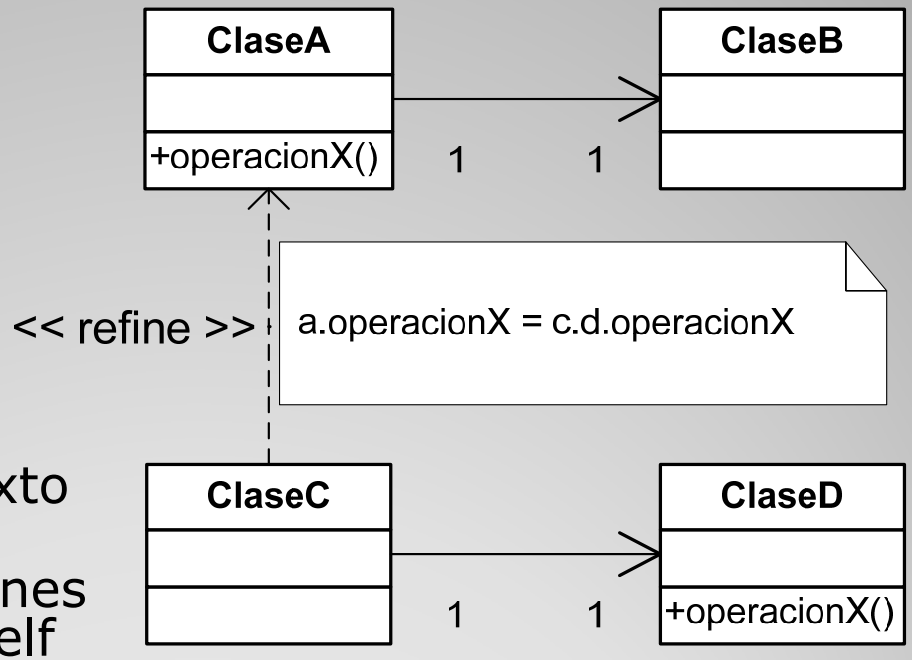
Catálogo de reglas

- Renombrar operación
  - Una operación cambia de nombre luego de la refactorización.
- Cambios OCL
  - Cada ocurrencia de la operación se reemplaza por el nuevo nombre.



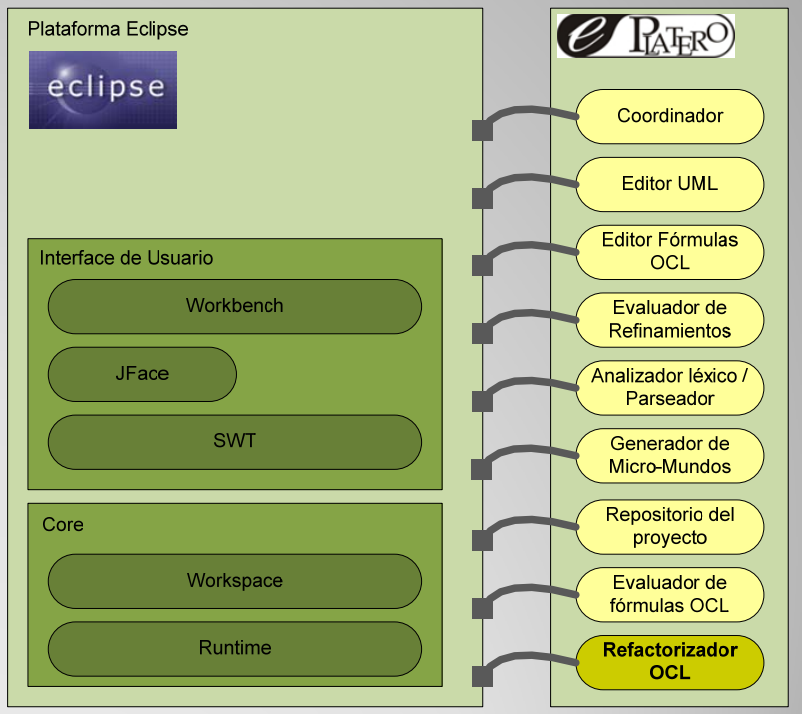
Catálogo de reglas

- Mover operación
  - Una operación es movida a una clase diferente luego de la refactorización.
- Cambios OCL
  - En las pre/post condiciones, el contexto debe ser cambiado.
  - Las pre/post condiciones debe ser redefinido self para referirse a la clase original.
  - Las referencias a la clase refactorizada deben ser reescritas para adaptarse a la nueva ubicación.



ePlatero

- Es una herramienta CASE de soporte al diseño dirigido por modelos
- Construido sobre la plataforma Eclipse
- Extensible mediante plugins

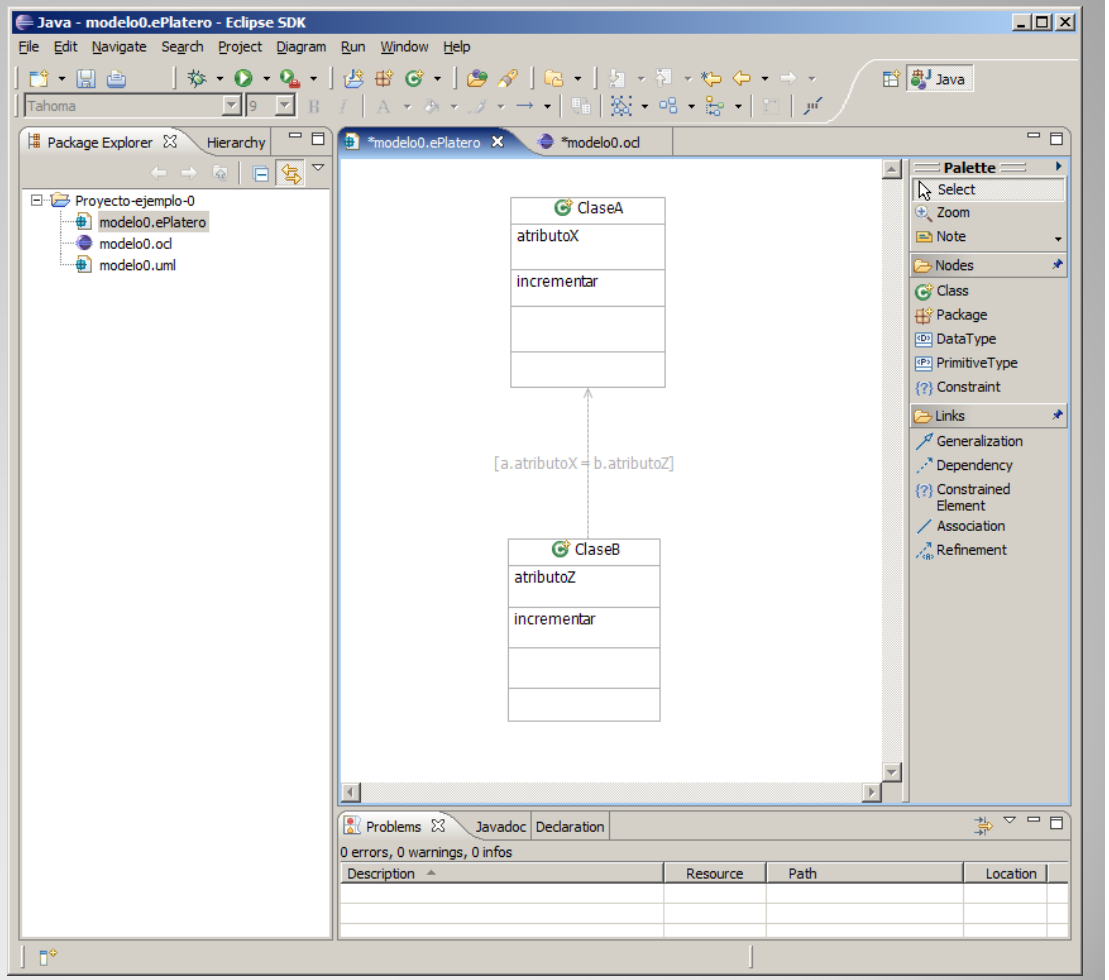


Módulo de refactorizaciones

- Nuevo plugin agregado a ePlatero
- Utiliza se complementa con otras funcionalidades existentes en ePlatero
  - Editor gráfico de modelos UML
  - Editor de restricciones OCL
  - Refactorizador OCL
    - Utiliza modelos UML
    - Modifica las restricciones OCL

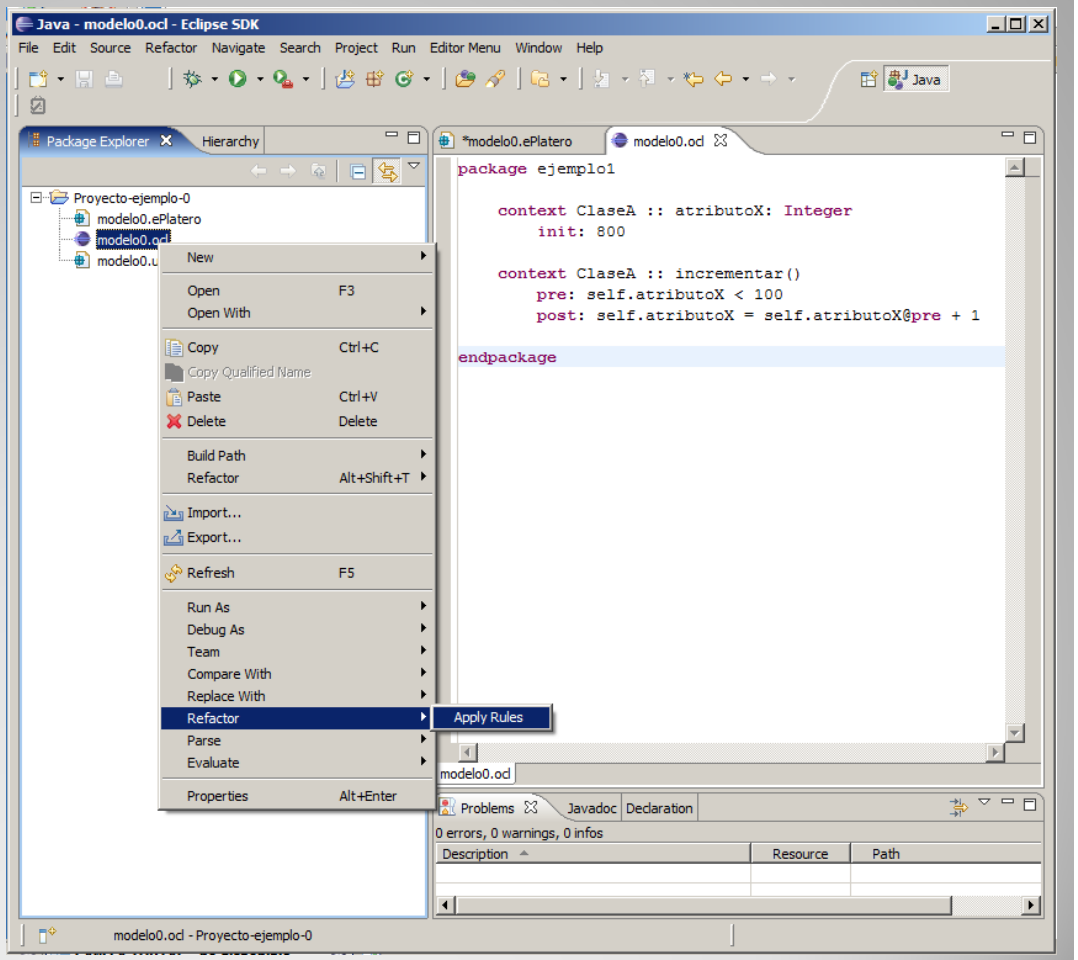
Modulo de refactorizaciones

- Diagramas UML en ePlatero



Modulo de refactorizaciones

- Integración con el editor OCL
- Menu contextual para acceso al modulo
  - Utilización de modelo UML
  - Utilización de restricciones OCL



Modulo de refactorizaciones

- Diseño interno
  - Punto de entrada: OCLRefactorFacade
    - Interface simplificada de la funcionalidad
    - Implementado con el patron Singleton
    - Coordina las acciones a realizar durante la refactorización

Modulo de refactorizaciones

- Pasos durante la refactorización
  - El proceso se inicia a partir del archivo OCL seleccionado.
  - Se instancia el metamodelo OCL a partir del archivo OCL.
  - Se instancia el metamodelo de UML a partir del diagrama UML seleccioado.
  - Del metamodelos UML se extrae el mapeo de la refactorización.
  - Del mapeo de la refactorización, se infieren las reglas.

Modulo de refactorizaciones

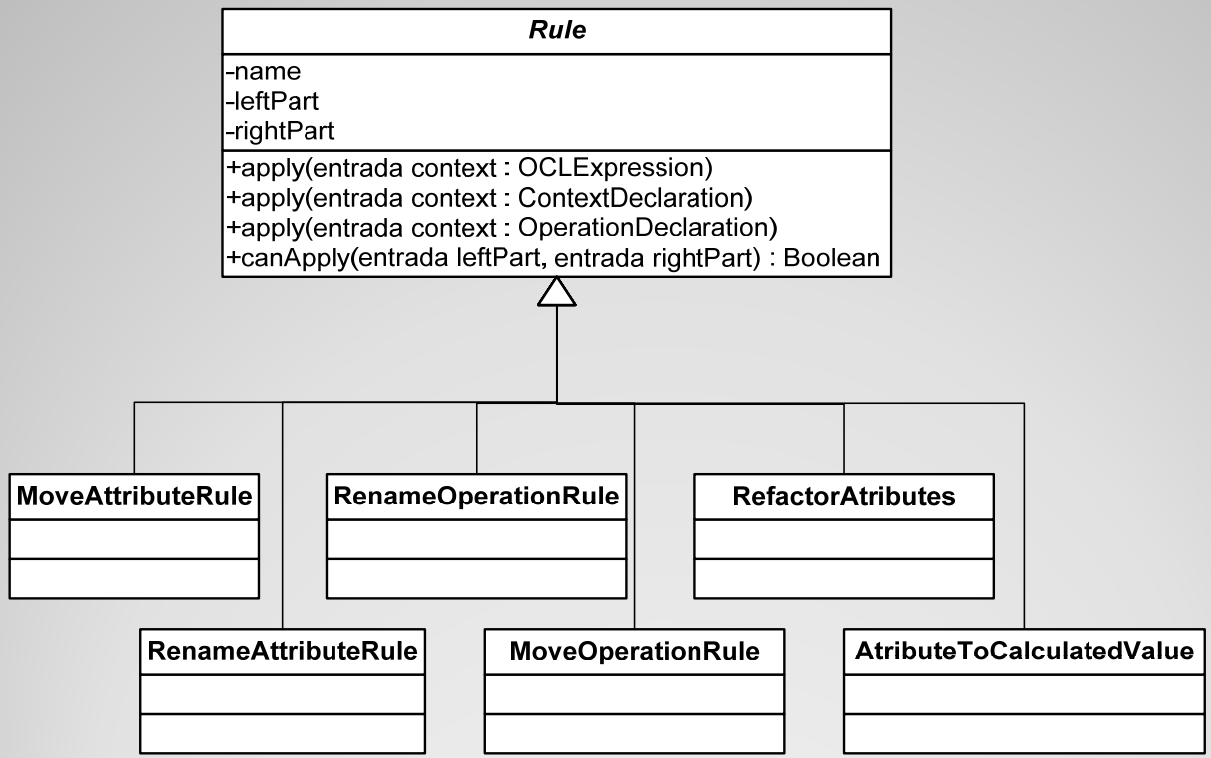
- Pasos durante la refactorización
  - Se instancia el conjunto de reglas.
  - Se aplican las reglas sobre el metamodelo OCL, mediante un Visitor. Este recorre el metamodelo OCL.
  - Por cada elemento del metamodelo que el visitor recorre, se ejecutan las reglas.
  - El orden de ejecución de las reglas se determina con un strategy.
  - Se regenera el archivo OCL con las reglas refactorizadas.

Modulo de refactorizaciones

- Reglas de refactorización
  - Clase abstracta con funcionalidades comunes
  - Clases concretas para cada una de las reglas
  - Las reglas se instancian a partir del mapeo de la refactorización, del modelo UML
  - Cada regla determina si puede aplicarse sobre cada mapeo de la refactorización
  - Cada regla implementa la refactorización de las restricciones OCL.

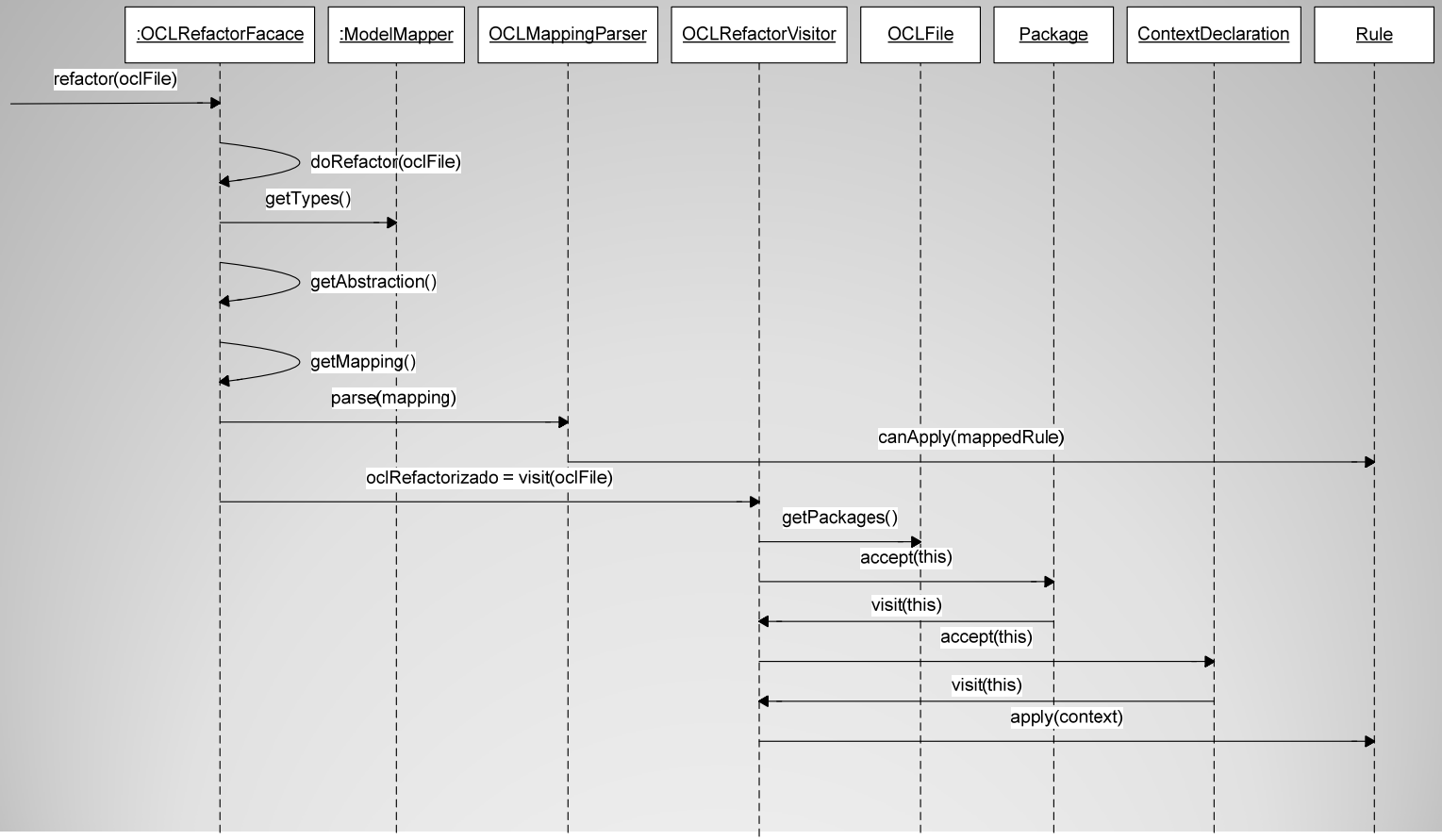
Modulo de refactorizaciones

- Diagrama de clases, reglas de refactorización



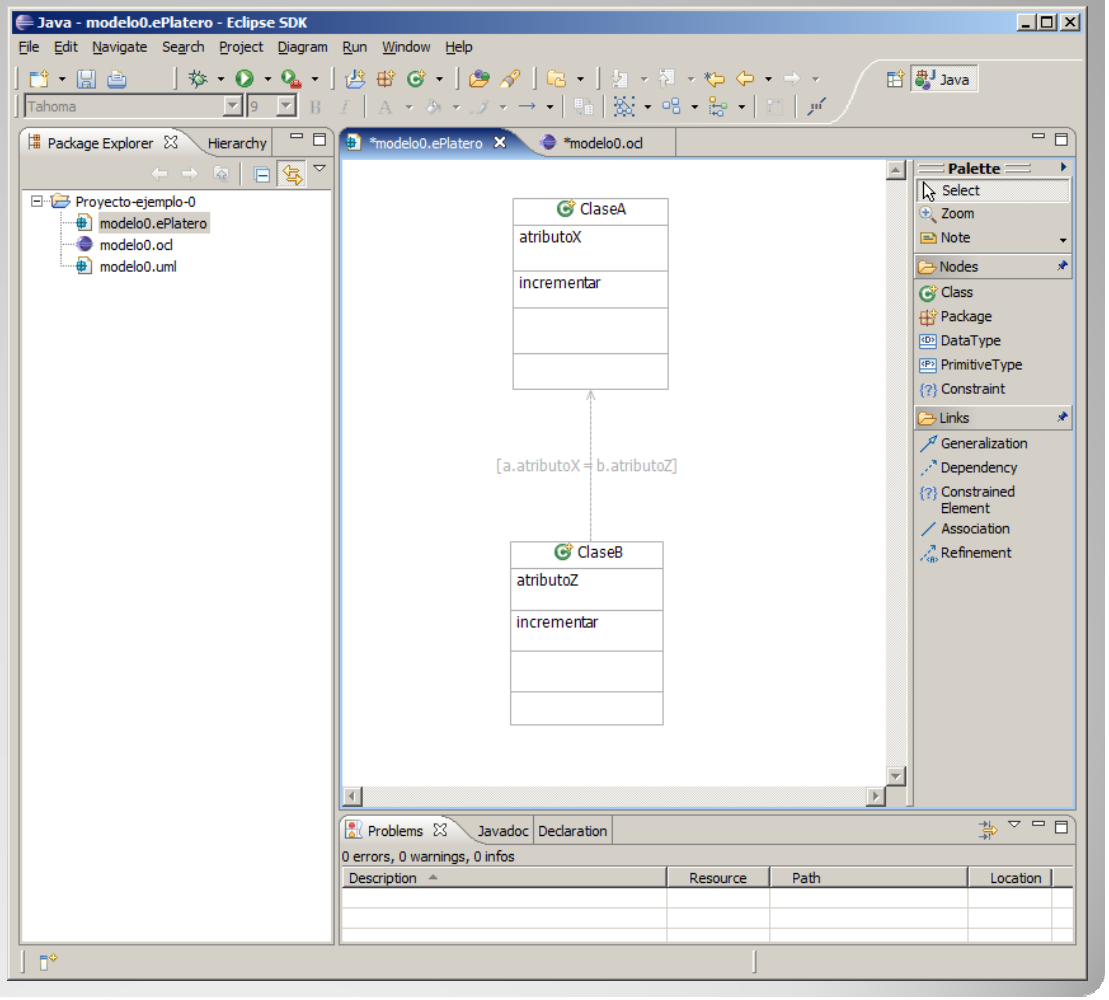
Modulo de refactorizaciones

- Diagrama de secuencia



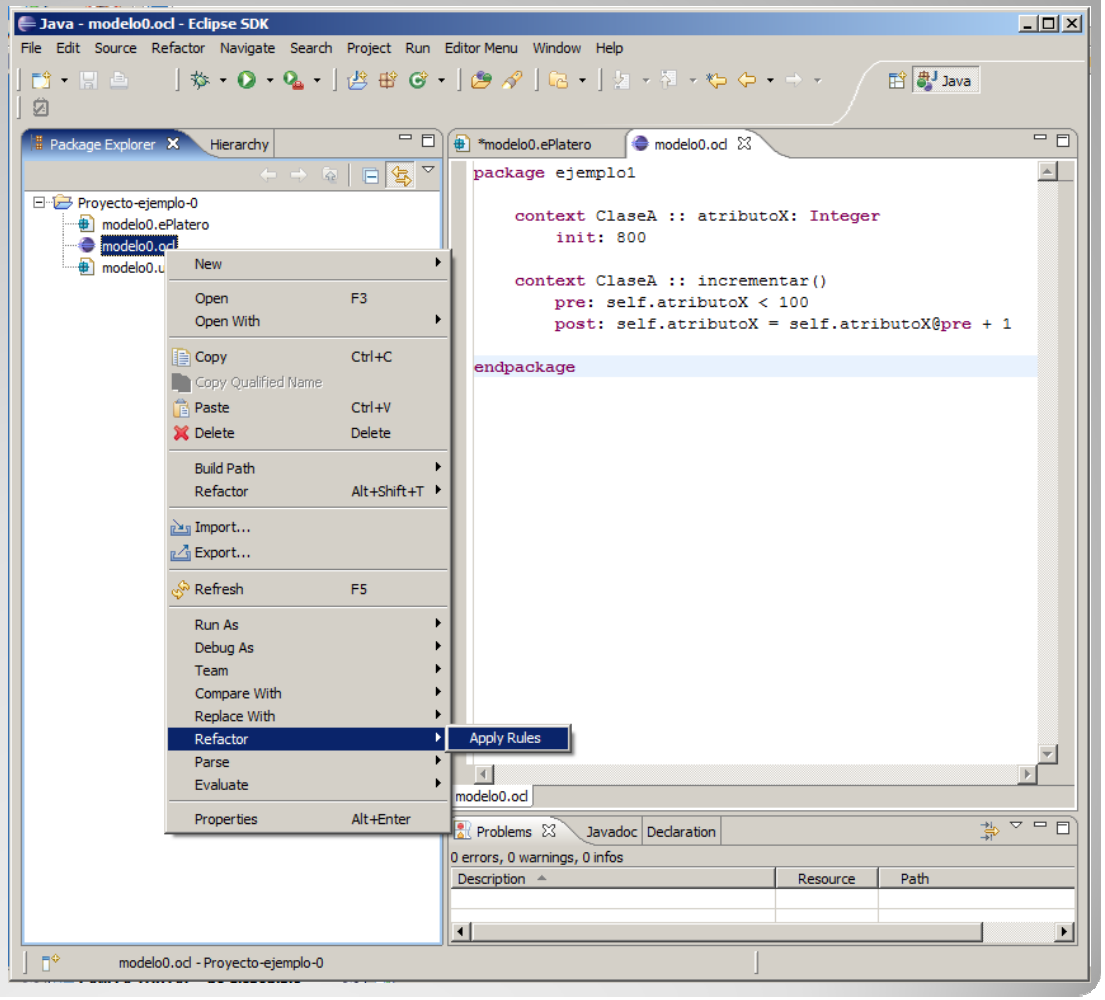
Ejemplo de uso

- Renombrar atributo
  - Modelo UML



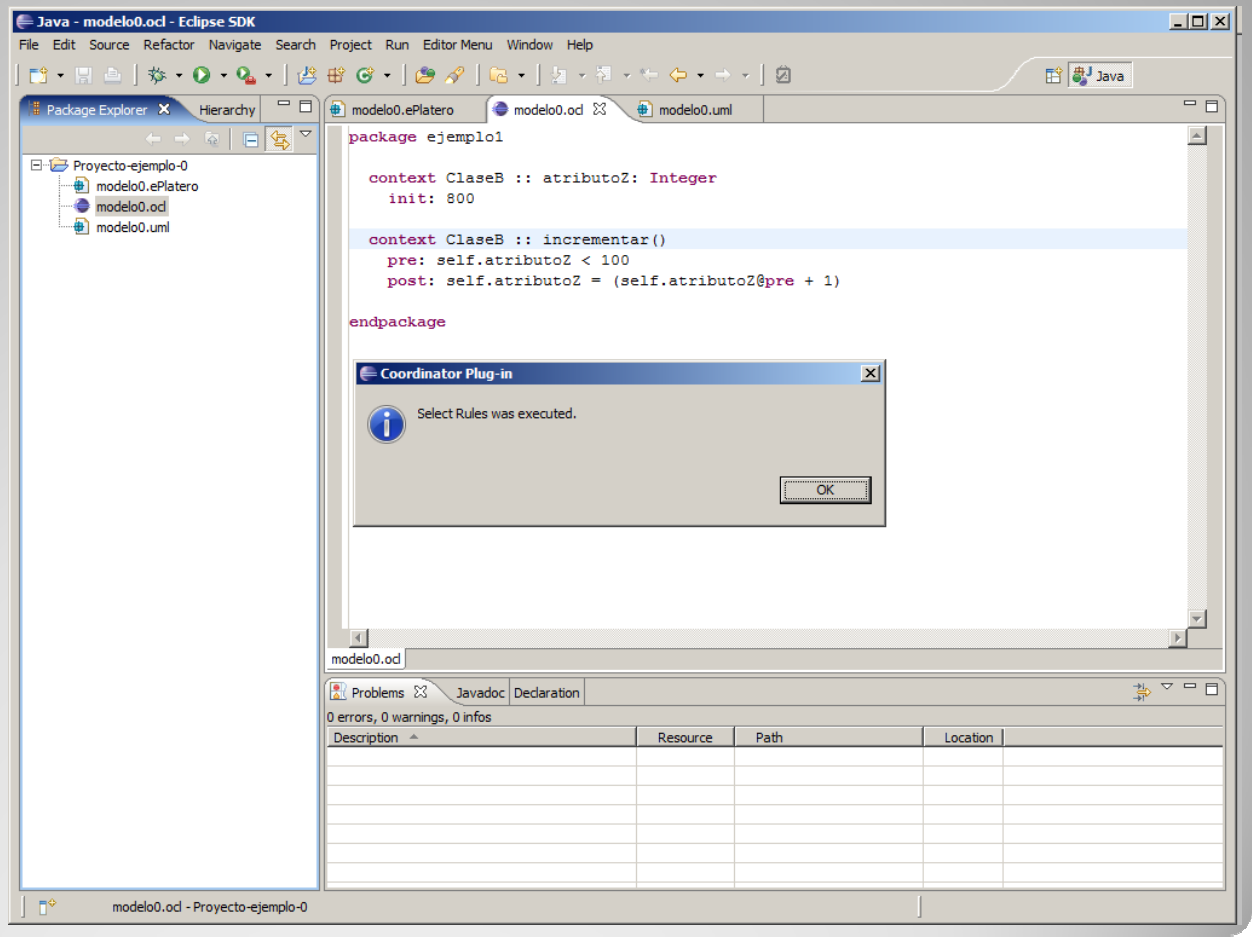
Ejemplo de uso

- Renombrar atributo
  - Restricciones OCL



Ejemplo de uso

- Renombrar atributo
  - Resultado final



Conclusiones

- MDA está cobrando fuerza los últimos años
- Mediante modelos y refactorizaciones de los mismos se pretenden combatir los problemas actuales del desarrollo de software.
- Con UML + OCL se consiguen modelos con el detalle suficiente para MDA
- Para que la industria adopte MDA, se necesitan herramientas adecuadas.

Conclusiones

- ePlatero es un buen ejemplo de herramienta de soporte a MDA
- En esta tesis se definió la forma de refactorizar diagramas UML con restricciones OCL.
- Se creo un catalogo de reglas de refactorización del código OCL
- Al aplicar las reglas se logra que los diagramas UML y restricciones OCL queden consistentes luego de las refactorizaciones

Conclusiones

- El catálogo es fácilmente extensible
- Se definieron algunas reglas
- Se sentaron las bases para la ampliación del catalogo en trabajos futuros.
- Se implementó una extensión de ePlatero que implementa la refactorización del código OCL.
- El plugin es fácilmente extensible al agregar nuevas reglas al catálogo

Fin del camino…

Muchas Gracias!!!