Universidad de Huelva

MFIS

Actividad Académica

ETSI

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

uhu.es

Autores:

- Ismael Da Palma Fernández

- Cristian Delgado Cruz
- Juan Jiménez Serrano
- Alejandro Pérez Domínguez

ÍNDICE

1. Introducción

Nuestra *ADD* consiste en el análisis general y del funcionamiento del programa **USE**, el cuál, será usado principalmente en las prácticas de la asignatura para el uso del ***lenguaje OCL***.

Además se realizará el mismo análisis de *diferentes programas* lo cuales posteriormente compararemos con **USE**, los programas propuestos son:

- **OCLARITY**
- **SIMPLEOCL**
- **iOCL**
- **Complemento OCL Eclipse**
- **USE**
- **OCL.js**
- **MagicDraw**

Nosotros indagaremos en los tres primeros, además del propio **USE**, el cual será con el que ***compararemos con todos los demás***.

El Lenguaje de Restricciones de Objetos (OCL) es un lenguaje de especificación textual que se utiliza en modelos UML para establecer reglas, restricciones y condiciones que no pueden representarse solamente con diagramas. OCL es una herramienta útil para mejorar la precisión y completitud de los modelos.

2. USE: Análisis y Funcionamiento.

2.1. Análisis

USE es un programa montado en *Java* sobre el lenguaje **UML**, para iniciarlo se utiliza el archivo por lotes llamado "*start_use*", tras ello, podemos comprobar que *se abre un terminal y una vista principal* donde podemos ver los submenús, el log, la ventana donde salen los archivos y una abajo donde salen como se implementó en el fichero usado, vease la imagen general en la figura 1.0.

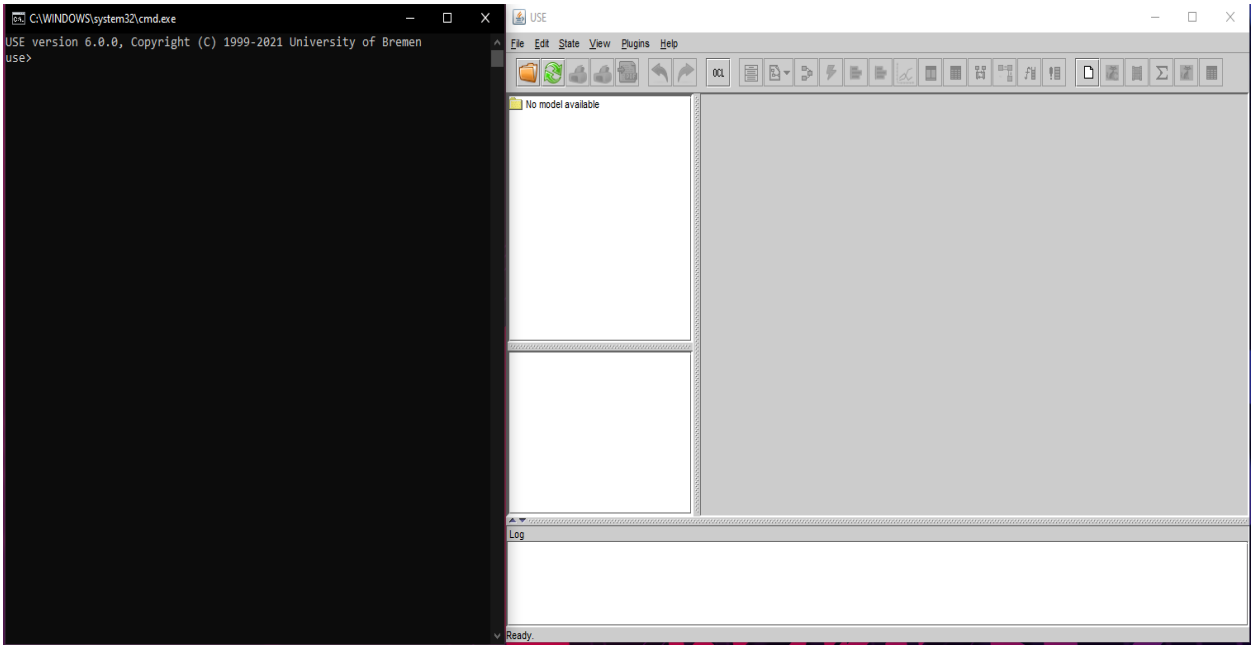

figura 1.0

Utilizaremos un ejemplo de las transparencias de clase, al incluirlo podemos ver como cambia la ventana principal figura 1.1, también podemos comprobar que al elegir una clase nos enseña su implementación.

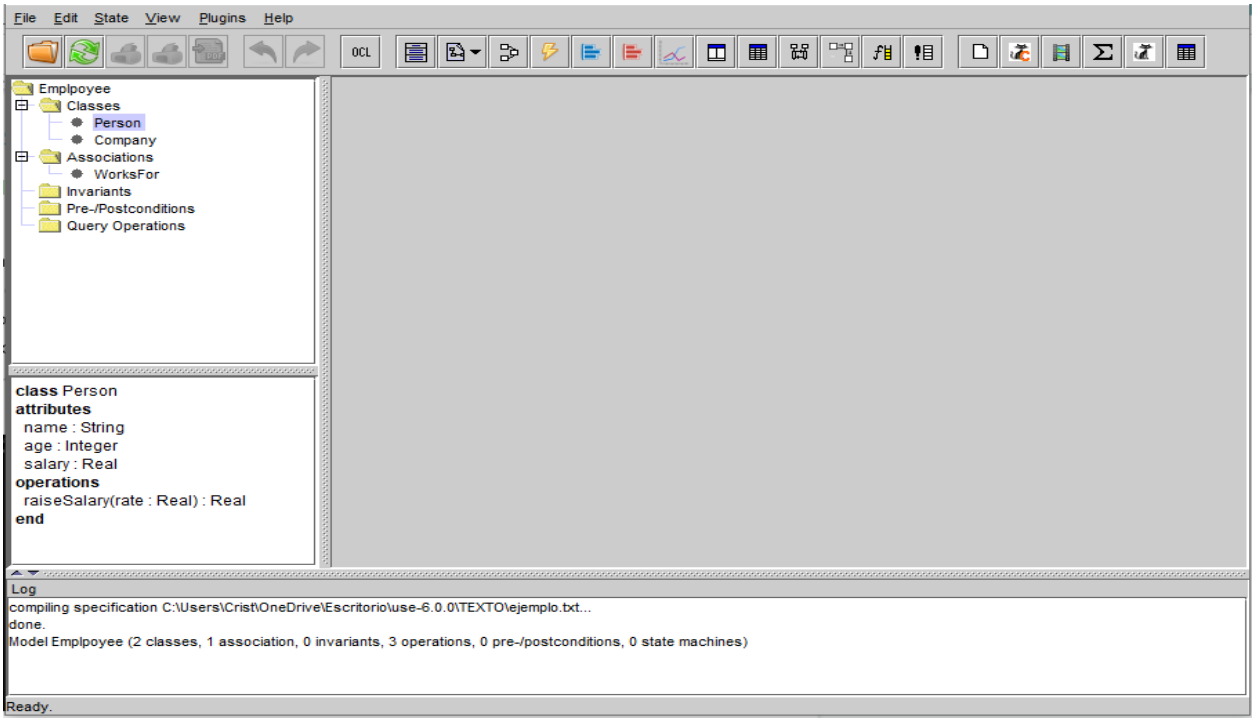

figura 1.1

Los menús en resumen se utilizan:

- **File:** abrir un nuevo *fichero* o guardar el que ya tenemos, además de exportarlo o imprimirlo.
- **Edit:** *deshacer o rehacer acciones.*
- **State:** crear objetos, crear estructuras, evaluar expresiones OCL, determinar estados, o resetearlos.
- **View:** nos *enseña* las diferentes vistas del programa como el diagrama de clases, de objetos, estados de la máquina, la pila etc.
- **Plugin:** es para validar el modelo , para transformarlo en varias cosas.
- **Help:** para ir al manual.

Utilizando los “botoncitos” de arriba podemos llegar a las *mismas opciones que en los menús desplegables* de una forma más visual véase en la **figura 1.3**

figura 1.3

Entre otras cosas en las opciones del menú *View* podemos ver el *diagrama de clases, el de objetos, cantidad de objetos, retocar los objetos, etc*; véase **figura 1.4**

Incluso podemos ver la *interfaz de comandos* que se utilizan al *insertar nuevos objetos y sus atributos*.

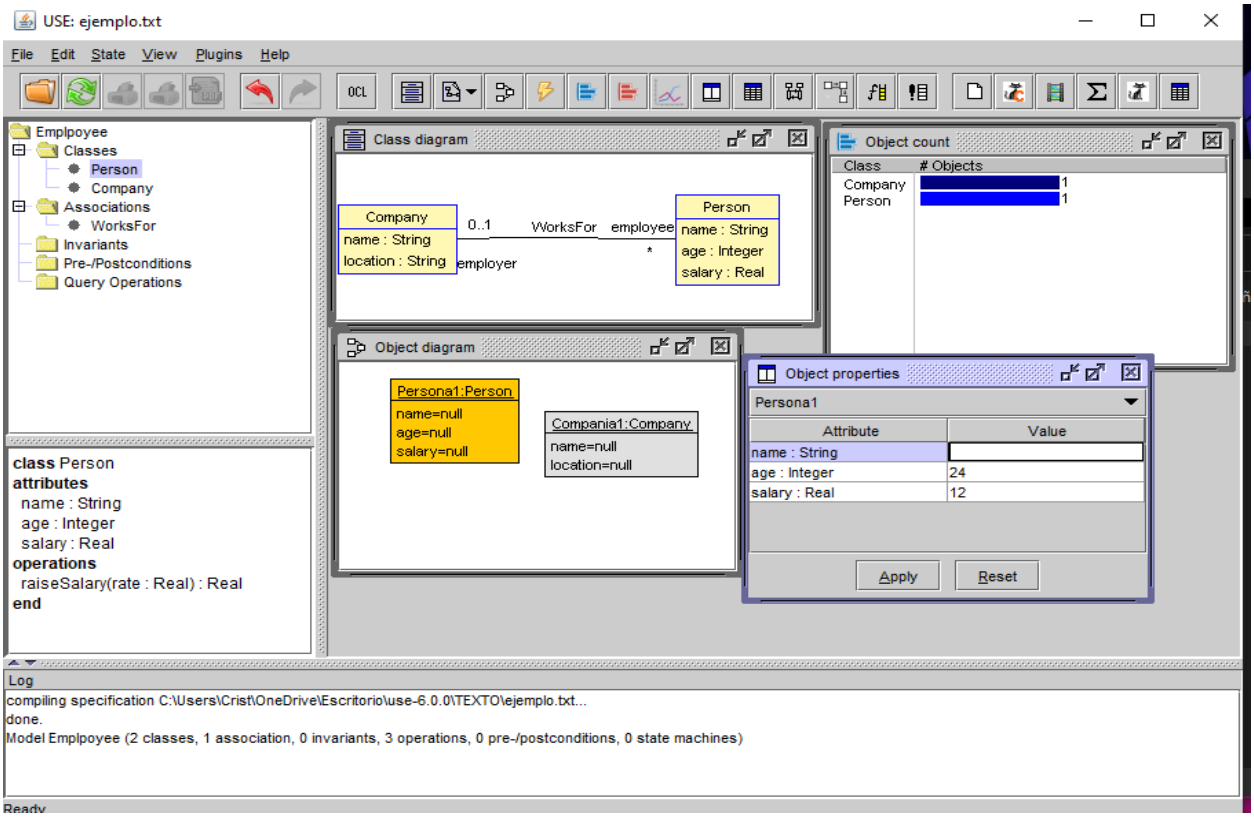

figura 1.4

2.2. Funcionamiento

Para la utilización del programa se necesita crear un fichero .use o en su defecto un **fichero .txt** que también lo detecta, para nuestro ejemplo hemos creado el siguiente txt:

```
model Emplpoyee

-- classes

class Person
attributes
   name : String
   age  : Integer
   salary  : Real
operations
   raiseSalary(rate : Real) : Real
end


class Company
attributes
   name : String
   location : String
operations
   hire(p : Person)
   fire(p : Person)
end

-- associations

association WorksFor between
   Person [*] role employee
   Company [0..1] role employer
end
```

Esta es la forma más simple de crearlo, ya que *se pueden crear* tipos *enumerados, clases abstractas, generalizaciones, etc.*

Los tipos que se pueden utilizar son **Integer** | **Real** | **Boolean** | **String** | **clase** y otros tipos como colecciones.

También se pueden declarar las *restricciones y contratos*:

```
context Orange
 inv OrangeInv: 1 = 1

context orange : Orange
 inv alwaysTrue: orange = orange
 inv: juice = true

context Lemon :: squeeze(i : Integer) : Integer
  pre: i>0
  pre lessThanTenOranges: i<10
  post alwaysTrue: true
```

en este ejemplo podemos observar varias restricciones y algunos contratos.

Además de todo esto, **existen muchísimas más funciones**, entre ellas *comprobar* si los objetos que vamos añadiendo *cumplen con las restricciones del modelo*, toda la demás información está en el manual original que está en *inglés*, pero tanta *la aplicación* como el *manual* están construidos de una manera muy *intuitiva* para que el uso sea sencillo.

3. OCLARITY

OCLARITY es una herramienta o entorno de creación basado también en OCL creado por ***Andreas Awenius.***

Actualmente se encuentra ***descatalogado***, o *en proceso de actualización* y la página web del proyecto se encuentra caída, la *poca información* que nos queda del mismo *es la siguiente*, figura 2.1

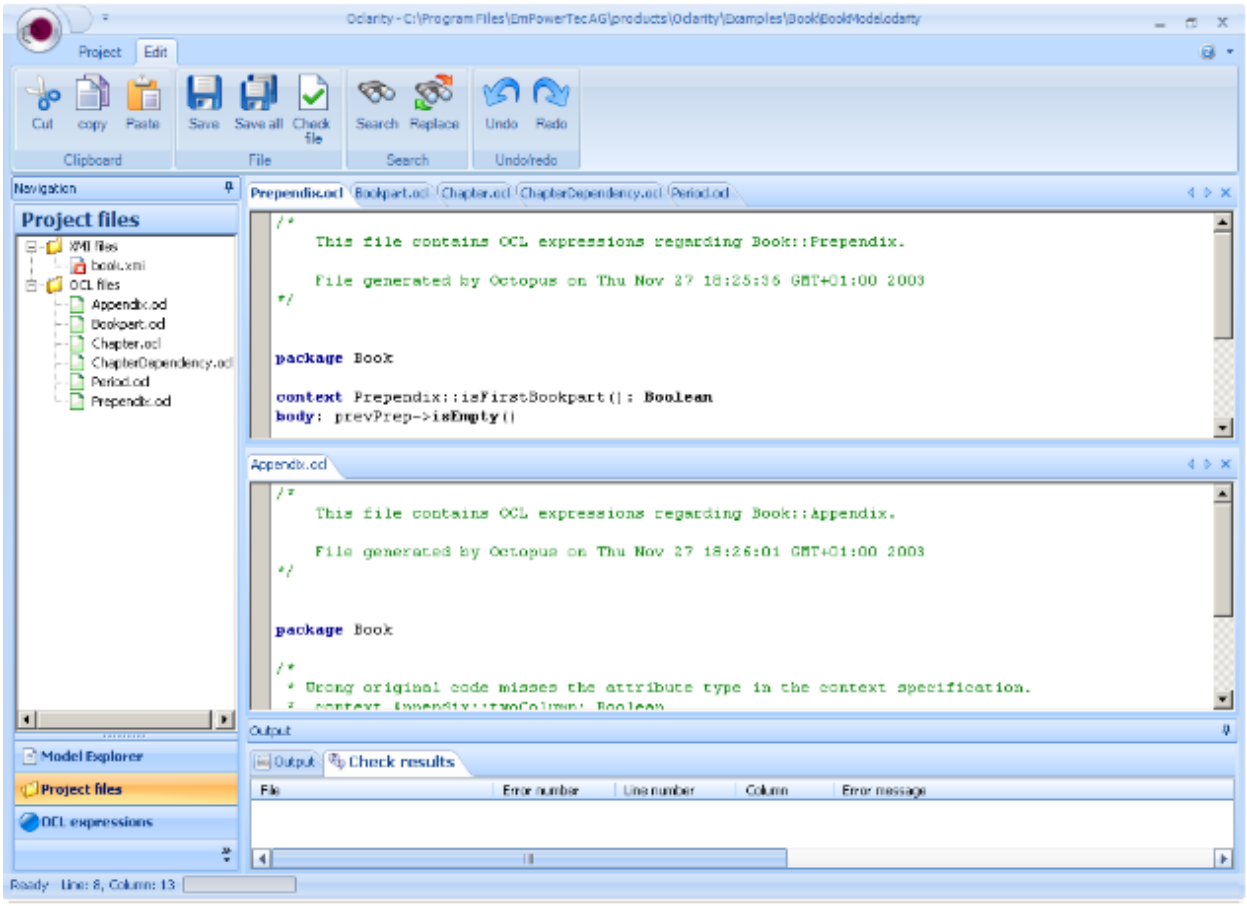

figura 2.1

Tenemos la **interfaz** donde se puede observar el panel con los directorios, la salida abajo, dos diferentes ventanas , la de arriba con las restricciones OCL y la de abajo con lo que parecen ser las clases.

En los menús de arriba se puede observar *botones* tales como *abrir, copiar, pegar, guardar, buscar, reemplazar, etc*.

Como no existen más páginas *no podemos saber cómo funciona completamente el programa*, pero al menos se puede ver que la interfaz es agradable e intuitiva.

4. SIMPLEOCL

4.1. Análisis

SimpleOCL es una implementación simplificada del lenguaje de especificación formal **OCL** que *se utiliza para describir restricciones y operaciones en modelos de objetos*, al igual que **USE**. **SimpleOCL** se desarrolló para ser más fácil de aprender y usar que la versión completa de OCL, y *se usa a menudo en el contexto de la enseñanza y el aprendizaje de OCL*.

SimpleOCL ofrece un **subconjunto reducido** de la sintaxis de **OCL**, pero sigue siendo capaz de expresar muchas de las restricciones y operaciones comunes que se requieren en los modelos de objetos. Esto lo hace una herramienta *útil para los principiantes* que están aprendiendo a trabajar con OCL.

SimpleOCL utiliza la interfaz de eclipse, ya que se trata de una *extensión (plugin) de esta misma aplicación* **figura 3.1**

El creador de esta herramienta es ***Dennis Wagelaar,*** este indica que brinda el soporte para importar gramáticas de otros lenguajes, ya que usa su propio compilador.

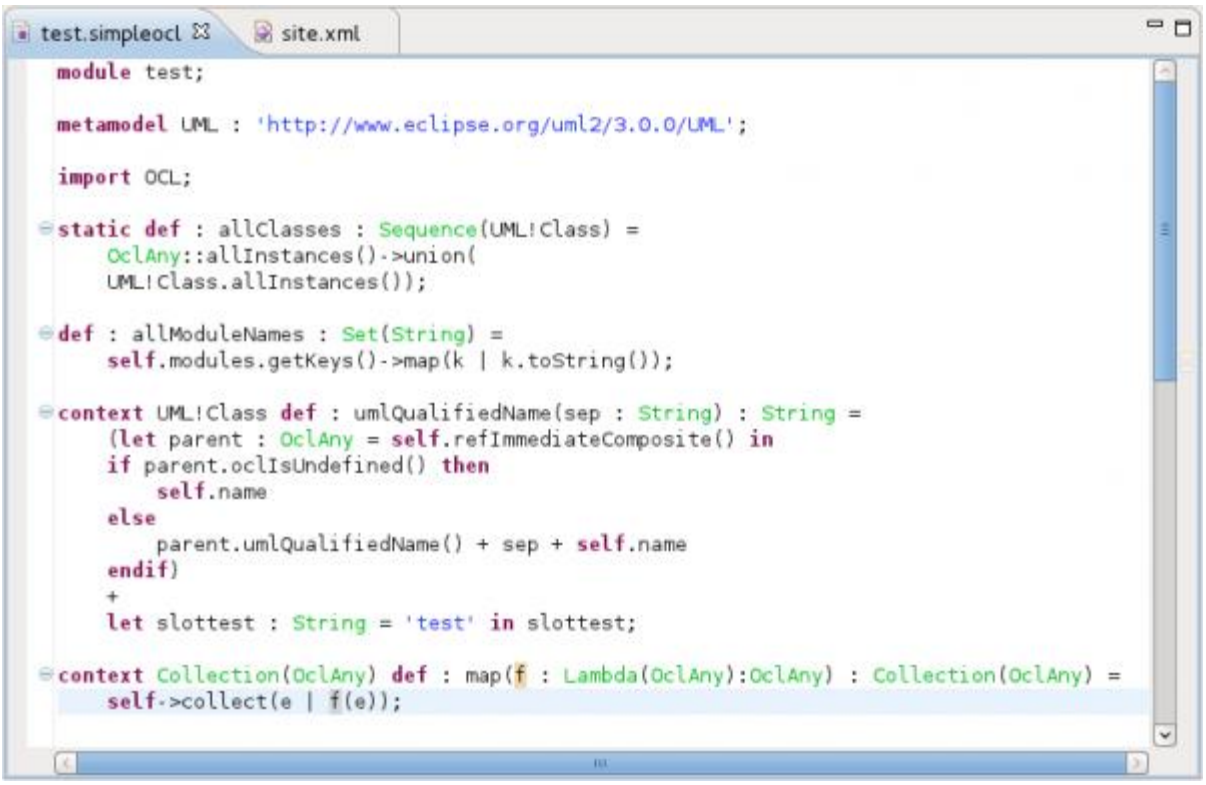

figura 3.1

4.2. Funcionamiento

La extensión funciona de forma sencilla, una vez instalada, se pueden *crear archivos de OCL* en eclipse, simplemente *importando OCL y eligiendo un metamodel*, funciona como el propio eclipse, definiendo clases módulos, teniendo su propio lenguaje de definición, basándose en un lenguaje de definición de mapeos (map).

Su forma de construirse es algo más complicada de lo normal pero en esencia utiliza la declaración de **context - inv de ocl**.

Aquí tenemos un ejemplo creado en github que refuerza la imagen anterior https://github.com/dwagelaar/simpleocl

5. iOCL

5.1. Análisis

Es una herramienta que permite a los desarrolladores de software interactuar con OCL de manera más dinámica y exploratoria. El **OCL interactivo** se utiliza a menudo para depurar modelos de objetos y verificar si las restricciones de OCL se cumplen correctamente.

El **OCL interactivo (iOCL)** es una consola o entorno de línea de comandos donde se pueden escribir expresiones OCL y obtener resultados en tiempo real. Los desarrolladores pueden escribir consultas OCL para consultar los modelos de objetos, evaluar las restricciones de OCL y probar diferentes escenarios. También pueden ver la estructura de los modelos de objetos y cómo se relacionan entre sí.

Es una herramienta muy útil para los desarrolladores que trabajan con OCL, ya que les permite experimentar con diferentes expresiones y escenarios **sin** tener que **modificar el código fuente** del programa.

Sin embargo, la información sobre esta herramienta es escasa, ya que los propios creadores han privatizado la mayor parte de esta.

5.2. Funcionamiento

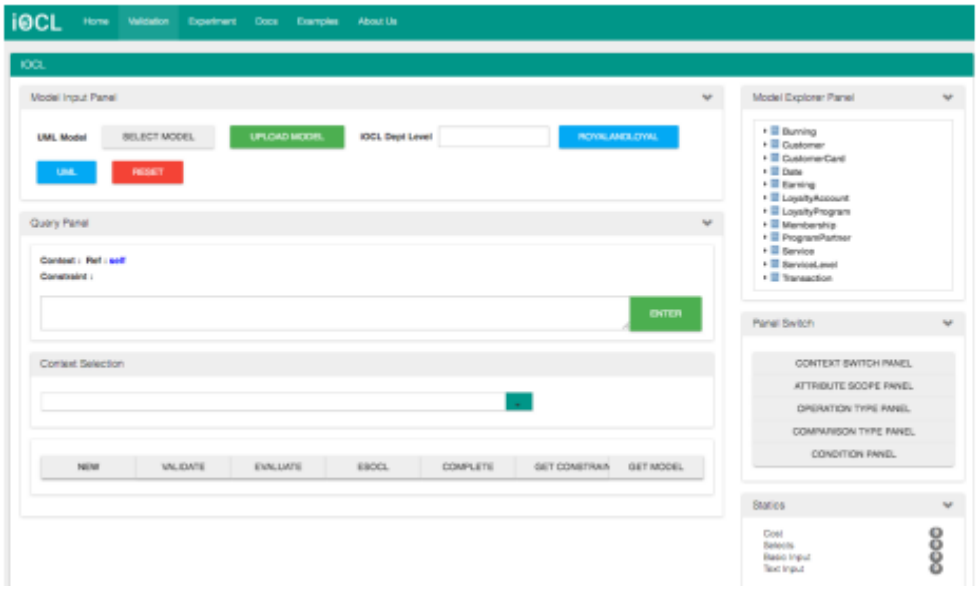

figura 4.1

Se puede **crear un sistema** desde cero, **o importarlo** desde un fichero .uml, el programa *mostrará* la siguiente interfaz, **figura 4.2**, donde se podrá añadir restricciones, o observar el diagrama de clases, los atributos , métodos etc.

Por otra parte tal y como podemos ver en la **figura 4.1**, se elige el contexto y *cuando uno selecciona el contexto*, iOCL automáticamente *muestra una lista de opciones*: Invariable, Precondición, Postcondición, que se puede elegir *para crear la restricción que nosotros queramos.*

También permite al usuario el uso de elección de donde se encuentra el atributo que quiere comparar en la restricción, por ejemplo si elige local, mostrará la siguiente tabla **figura 4.3**

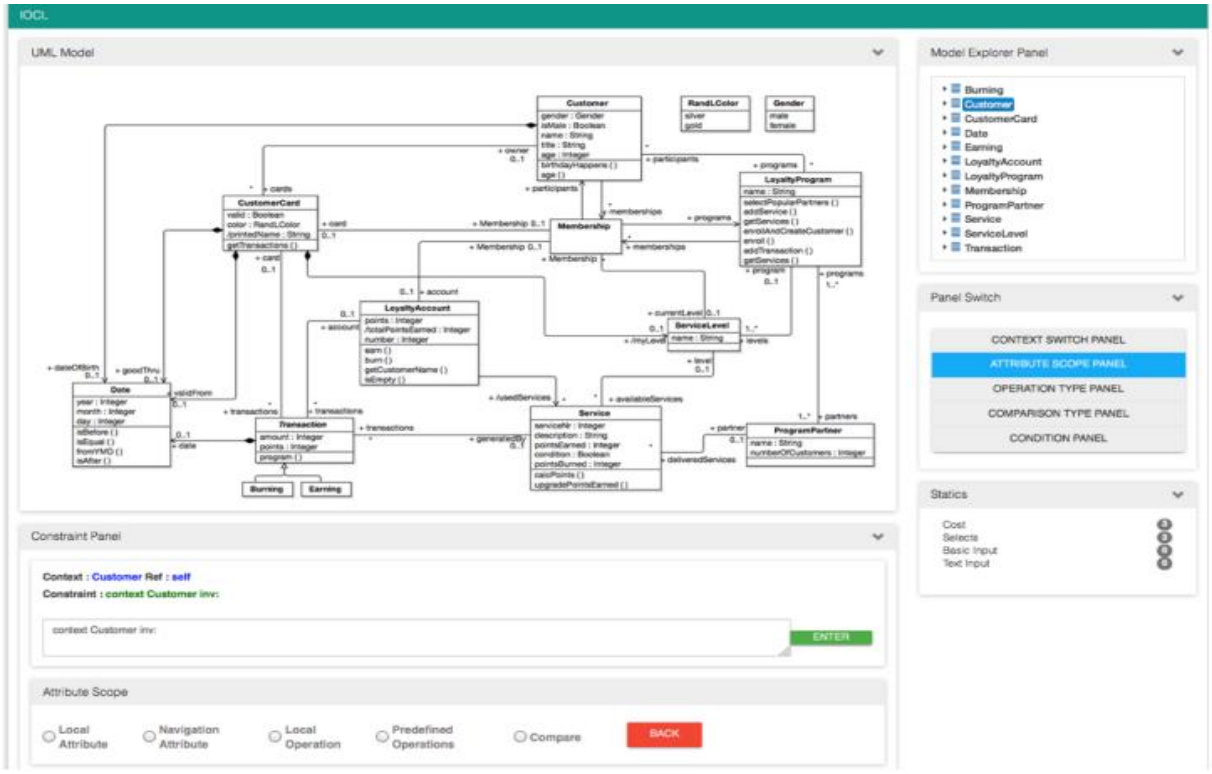

figura 4.2

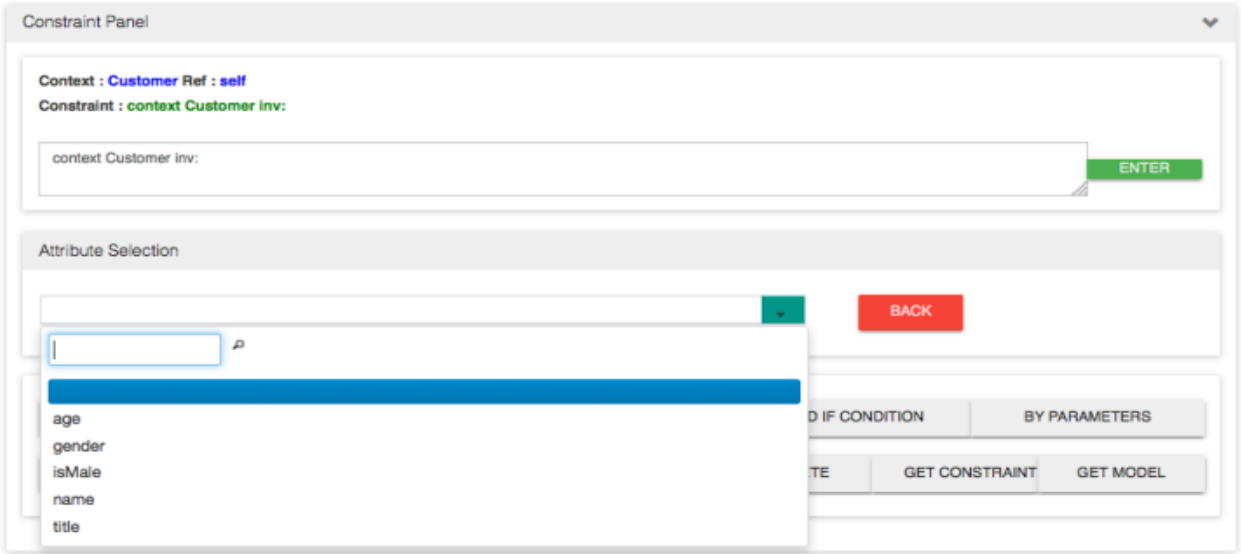

figura 4.3

iOCL avisará al usuario de si hay algún problema con la sintaxis de la restricción y *si no* la hay el usuario podrá *validar el caso* terminando el programa mostrando un mensaje como el siguiente, sea falso o verdadero la restricción que queremos comprobar con los atributos e instancias que se hayan declarado. **figura 4.4**

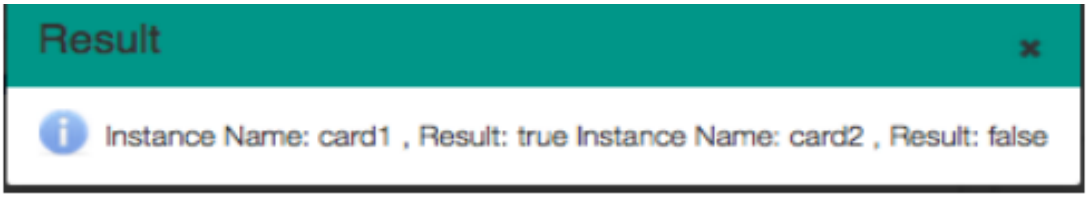

figura 4.4

6. Comparativa de las herramientas

Veamos un resumen de las principales características de cada uno de los programas que vamos a comparar con USE:

- **OCLARITY:** *no hay información sobre esta herramienta.*
- **SimpleOCL:**
  - Implementación **simplificada** de OCL.
  - Es capaz de expresar muchas restricciones y operaciones pese a su simplicidad.
  - Útil para los **principiantes** para *aprender a trabajar con OCL*.
  - No es un programa en sí, sino un **plugin**.
  - Tiene su **propio compilador**.
- **iOCL:**
  - Interacción **más dinámica y completa** con OCL.
  - No es ni un programa ni un plugin, sino una **página web**.
  - Los resultados de las expresiones y operaciones son **en tiempo real**.
  - Permite experimentar con **diferentes escenarios** sin tener que modificar el código fuente.

En términos de simpleza o intuitividad, el más sencillo de usar es **iOCL**, ya que *basa todo su interfaz gráfica en tablas y esquemas,*

siendo mucho más fácil de entender, y de utilizar, dando al usuario una experiencia plena de el uso de **UML** sin tener que saber realmente utilizar el lenguaje, por otra parte, *si ya estás versado* en el lenguaje **UML**, quizá la herramienta **iOCL** se queda corta en cuanto a contenido, ya que no permite indagar más de lo que las tablas y esquemas lo permiten en el propio lenguaje, así que, si tenemos en cuenta esto, ***USE*** *aparenta ser el que mejor equilibrio tiene entre dificultad y utilidad*, ya que permite al usuario un control más completo de las clases, instancias y restricciones del modelo sin llegar a complicar tanto el lenguaje *como SIMPLEOCL.*

Cambiando la perspectiva, basando nuestro puntaje en la portabilidad que tiene la herramienta que utilizamos, podemos dejar claro que **SIMPLEOCL** sería, sin duda, la mejor, ya que ***permite exportar y importar en numerosos lenguajes*** y *funciona en todas las máquinas que tengan instaladas o puedan tener*, **Eclipse**.

También hay que tener en cuenta que **las únicas dos herramientas utilizables** o alcanzables a día de hoy son **USE y SIMPLEOCL** debido a que las otras dos se encuentran descatalogadas, dado que no se puede acceder a ellas de ninguna forma, ya que *sus páginas no existen o han sido cerradas*. Por ese motivo, no se menciona en ningún momento a **OCLARITY**, ya que es imposible de acceder de ninguna forma a información si quiera, *al no tener funcional su página web, la cual era la fuente de información y portal de descarga de la misma.*

VENTAJAS/ HERRAMIENTAS	USE	OCLARITY	SIMPLEOCL	iOCL
Intuitividad				✘
Completo	✘	✘		
Disponibilidad	✘		✘	✘
Interfaz Gráfica	✘	✘		
Portabilidad	✘	✘	✘	✘
Manuales de uso	✘		✘	✘
Gratis	✘		✘	

En conclusión USE es la mejor herramienta, al menos para empezar, por su *equilibrio de simpleza y gran aporte de información*, siendo **SIMPLEOCL** el siguiente paso en *alguien que quiere indagar más* allá sobre el lenguaje UML y sus variantes.

7. Bibliografía

Página con información de OCLARITY

Página con información de SimpleOCL

Página de descarga de SimpleOCL

Página con información de iOCL