

# Extensiones al ambiente de *Smalltalk* usando *AOP*

Alejandro Alvarez

LIFIA  
Facultad de Informática - UNLP  
La Plata - Argentina  
*alejandro.alvarez@lifa.info.unlp.edu.ar*

5 de noviembre de 2007

## 1. Area

Reportes de Experiencias de Desarrollo o Investigación.

## 2. Abstract

### 2.1. Introducción

Alterar un ambiente de programación como *Smalltalk* ya sea para agregar o modificar funcionalidad pre-existente, es una tarea que demanda tener ciertos cuidados al codificar; esto se debe a que el código que se está ingresando o modificando está siendo utilizado por objetos del ambiente. Si se ha cometido un error de programación es muy probable que el ambiente quede inestable, y dependiendo del tipo de error, la recuperación del sistema podría no ser posible. Ante esta situación se corre el riesgo de perder los cambios realizados con anterioridad. La propuesta que se presenta es la de hacer uso de la programación orientada a aspectos para incorporar el nuevo comportamiento.

### 2.2. Identificación del problema

Se supone que se quiere modificar la forma en que se construye un menú, para simplificar el ejemplo, se tendrá en cuenta que sólo se debe modificar un método para llevar a cabo éste objetivo. Una vez hecha la modificación necesaria, y según nuestro punto de vista realizada correctamente, se procede a aceptar el cambio. La próxima vez que un menú necesite ser construido nuestro método modificado será invocado para ser ejecutado, como consecuencia de esto, si se ha cometido un error de programación, al querer construir un menú, el error probablemente causará una excepción que derivará en la ejecución del *debugger*, que como sabemos también contiene menús.

Esto deriva en que se tiene un “*error recursivo*”, poniendo en peligro la recuperación del ambiente y la integridad de la imagen. Como consecuencia si se termina la ejecución del ambiente se perderán los últimos cambios.

### 2.3. Solución propuesta

La idea es la de codificar el comportamiento de manera aislada para luego mediante el uso de aspectos poder incorporarlo. Para ilustrar este caso, se implementó una mejora al ambiente que se denominó *Modern Menus*.

El objetivo de la mejora fue proveer al usuario del ambiente con menús un tanto más modernos, de aquí el nombre dado a la implementación. La funcionalidad que se agregó fue la de mostrar en cada menú sólo los  $n$  ítems más usados, con dos posibilidades de orden para los mismos, configurables por el usuario. La primera es que los ítems estén ordenados según su frecuencia de uso y la otra es que se conserve su orden original.

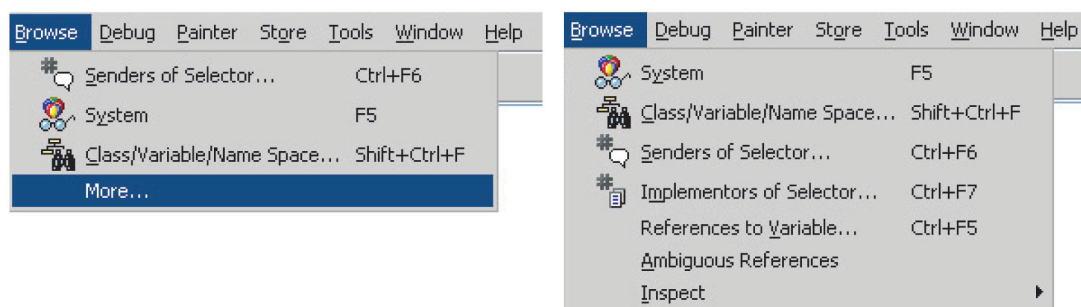


Figura 1: Un menú contraído (izq.) y luego redesplegado (der.) al seleccionar el ítem *More...*

Para esto se utilizó *AspectS*, que es una implementación que permite trabajar con *AOP*, en particular en *Squeak* y *VisualWorks*. Está basado en el modelo de *AspectJ* y hace uso de la metaprogramación para introducir el comportamiento de aspectos. A diferencia de *AspectJ*, el proceso de *weaving AspectS* lo realiza dinámicamente en tiempo de ejecución haciendo uso de *method wrappers*, más precisamente de *block method wrappers* que permiten introducir bloques de comportamiento. El proceso de *weaving* es ejecutado cada vez que un aspecto es instalado, introduciendo los *block method wrappers* en los *MethodDictionaries* según lo especificado en los *join points* de cada *advice* definido por el aspecto.

### 2.4. Conclusión

De lo expuesto anteriormente se concluye que la utilización de *AOP* para el desarrollo de extensiones al ambiente de programación de *Smalltalk* facilita la recuperación en situaciones de error. Por otro lado, la aplicación de *AspectS* permitió separar concretamente el *concern* referente a las mejoras en el ambiente, aumentando notablemente su modularidad respecto de otros componentes del ambiente. También es posible activar o desactivar la funcionalidad en tiempo de ejecución y no se pierde la implementación original.