

## **Trait aware refactorings**

### **Extendiendo la utilidad del refactoring browser**

Sección de Investigación/Educación

Alejandro González. [mrgonza78@gmail.com](mailto:mrgonza78@gmail.com)

Hasta el momento los refactorings disponibles en cualquier refactoring browser de cualquier dialecto Smalltalk solo operan en elementos típicos del ambiente: clases, métodos, jerarquías de clases, variables de instancia, etc. Pero hace algún tiempo un nuevo elemento apareció en el paradigma de objetos y en particular en Smalltalk: traits.

Traits son construcciones lingüísticas que permiten que un grupo de métodos puedan, como conjunto, ser nombrados y reutilizados en cualquier parte del ambiente, sin importar las jerarquías de clases en donde se las use. Con esta nueva construcción, las clases no solo pueden definir y heredar métodos sino que también pueden incorporar rasgos o características nuevas usando los métodos provistos por uno o más traits.

Puesto en otros términos un trait es un rasgo, una característica o propiedad que comparten distintas clases de objetos. Por ejemplo, los números son elementos que pueden compararse, por consiguiente un rasgo, característica o propiedad de los números es el hecho de ser comparables. Pero este rasgo, característica o propiedad nos es exclusiva de los números, las fechas también las comparten al ser comparables entre si. Y lo mismo ocurre, por ejemplo, con las distancias, los volúmenes y otras magnitudes.

De esta forma los traits permiten un nuevo estilo de programación en la cual estos se convierten en la forma primaria de reuso de código, en vez de ser las clases.

Contar con la disponibilidad de traits tomo tanta relevancia que Squeak ya los incluye en sus distribuciones base y, por ejemplo, ya existen ports para disponer de este nuevo estilo de programación en Visual Works.

El problema existente es la falta de herramientas que soporten este nuevo estilo de programación, haciendo que la programación y navegación de traits se realice en forma casi manual. En particular no existe soporte para la refactorización "towards, to and away from" traits. El refactoring browser solo incluye refactorings típicos como por ejemplo "Push up method" y "Rename method".

Es decir, si tuviésemos un sistema de objetos compuesto solo por clases (además de objetos, pero no por traits), y como programadores tuviésemos la posibilidad de poder mejorar el diseño refactorizando mucho del código repetido en un conjunto de traits que luego podrían ser reutilizados en cualquier parte de la jerarquía, entonces deberíamos crear los traits apropiados, copiar métodos de las clases a estos traits, luego borrarlos de las clases y por ultimo hacer que las clases usen los traits en cuestión. Todo este proceso se convierte en una tarea repetitiva, tediosa y muy propensa a errores si tuviésemos que realizar tales refactorizaciones en forma manual.

Contar con una herramienta de refactoring con soporte para traits sería de mucha ayuda para poder aplicarlos en sistemas que aún no los usan y podría ayudar a acelerar, en un factor considerable, etapas exploratorias en donde a priori no se

sabe si la inclusión de traits en el sistema podría traer beneficios o no. Muchos de los trabajos e investigaciones acerca de traits, están relacionados con la refactorización de jerarquías completas (por ejemplo Collection y subclases fueron totalmente refactorizadas con traits), y otras relacionados con las herramientas y metodologías que este nuevo estilo de programación acarrea. La mayoría de estas investigaciones remarca la necesidad de contar con herramientas que permitan realizar refactorizaciones con traits en forma automática.

La charla, que se presentaría en el marco de *investigación/educación*, tiene como objetivo presentar un nuevo conjunto de refactorings implementados en el refactoring browser de Squeak, que además de operar sobre los elementos ya conocidos, también operen con traits. Previamente a introducir estos nuevos refactorings, se hará una breve incursión acerca de traits y se ejemplificará su uso para facilitar la comprensión de este nuevo concepto.

Por último se hará *una presentación en vivo* en Squeak en la que se mostrará como se pueden aplicar estos refactorings para mejorar ciertos aspectos del ambiente refactorizando, por ejemplo, la clase Magnitude en un trait, y también refactorizando algunos aspectos de una librería llamada Acacocagua que reifica el concepto de medidas (volúmenes, distancias, velocidades, etc.).