

Reificación de refactorings en sistemas de software configuration management

Diego Tubello (dtubello@dc.uba.ar)

El diseño de software es una tarea difícil. Parte de los objetivos del diseño es que el software sea reusable y mantenible. Normalmente obtener software reusable requiere de varias iteraciones, en las que se producen cambios estructurales.

La técnica utilizada para realizar cambios estructurales en el sistema se la denomina *refactoring*. Los refactorings son cambios que modifican la estructura del programa sin afectar su comportamiento externo.

Aunque esta técnica se haya utilizado por mucho tiempo, el primer estudio formal fue realizado por William Opdyke en su tesis de doctorado *Refactoring Object Oriented Frameworks*. En este trabajo se propone una forma automática de llevarlos a cabo en software orientado a objetos. Se presentan una serie de refactorings que pueden ser aplicados automáticamente sin modificar el comportamiento externo del programa.

A partir de este trabajo se comenzaron a implementar herramientas para realizar refactorings automáticos en los distintos lenguajes. La primera en surgir (y todavía una de las más completas) fue el *Refactoring Browser* para Smalltalk. Posteriormente surgieron herramientas similares para otros lenguajes.

La aparición de estas herramientas hizo que los diseños pueden evolucionar más libremente, permitiendo hacer cambios estructurales que afecten gran parte del programa de forma automática sin introducir errores en el proceso.

Desarrollo en equipo y la relación con Refactorings

Cuando se desarrolla en equipo, las distintas líneas de trabajo se integran utilizando herramientas de *software configuration management* (SCM). Las herramientas de SCM manejan entidades que son versionadas para indicar que no pueden ser modificadas de ahí en adelante.

Algunos sistemas, como CVS, manejan archivos como entidades, y llevan registro de las modificaciones realizadas a los archivos que conforman el programa. Otros como ENVV manejan módulos, clases y métodos como entidades para el versionado. Estas herramientas poseen una línea base (última versión integrada) a la que se le impactan los cambios que realizan los programadores que trabajan en paralelo.

Si las distintas líneas de trabajo modifican las mismas entidades pueden producirse conflictos que no son fácilmente integrables automáticamente. Este problema se acentúa con los refactorings, debido a la facilidad que nos proveen los ambientes de desarrollo para realizarlos y la dimensión del impacto que suelen tener.

Con las herramientas de Refactoring, estos son tratados como un único cambio, por ejemplo, un renombre de una clase. Sin embargo, para el sistema de SCM ese mismo refactoring se ve como una serie de cambios distribuidos por gran parte del programa. Por ejemplo, al integrar un cambio hecho por un refactoring en los SCMs especializados para lenguajes de objetos veríamos que una clase fue borrada (la del nombre a cambiar), otra clase fue creada (con el nuevo nombre) y veríamos modificaciones aisladas en el cuerpo de los métodos que referenciaban esta clase. Si otra línea de trabajo hizo cambios sobre la clase que fue renombrada, no hay manera de

relacionar estos cambios con la nueva clase creada a partir del refactoring. Más aún como en realidad para el SCM ocurrió que una clase fue borrada y luego otra creada, se pierde la historia de la clase. Esto último sucede porque las entidades son referenciadas por nombre, al cambiar el nombre, se transforma en otra entidad y no hay manera de relacionarla con la anterior. Cosas similares ocurren al renombrar métodos y al aplicar otros refactorings.

Más allá del caso patológico de los renombres, un refactoring, suele estar compuesto por muchos cambios muy distribuidos. Si los refactorings son una práctica cotidiana, la integración de las distintas líneas de trabajo con las herramientas actuales puede generar muchos conflictos que no permitan realizarla de forma automática.

Objetivo del trabajo

El objetivo del trabajo es agregar el concepto de refactoring a una herramienta de SCM, para lograr que la integración de los refactorings realizados en distintas líneas de trabajo paralelas puedan ser integrados en forma automática.

Se espera que gran cantidad de las integraciones que se debían realizar manualmente sean automáticas y en aquellos casos en los que no se pueda (dado que se tiene la información de los refactorings realizados) la integración manual sea más sencilla.

De esta manera la facilidad que nos proporcionan las herramientas de refactorings del lado del ambiente de desarrollo, la prolongamos a los procesos de integración realizados con las herramientas de SCM.

Esto sin duda reduciría drásticamente la cantidad de integraciones que no pueden ser realizadas de forma automática.

Entre los puntos más importantes del trabajo están:

- Llevar un registro de los refactorings realizados en una línea de trabajo para poder ser integrados a la línea base.
- Extender el concepto de refactoring para adaptarlo a la noción de distintas líneas de trabajo.
- Modificación de las herramientas de integración: en nuestro caso se extenderá una herramienta de integración automática desarrollada por Mercap S.R.L.