Análisis de Identificadores para Abstraer conceptos del Dominio del Problema

Javier Azcurra, Mario Berón

Facultad de Ciencias Físico Matemáticas y Naturales
Universidad Nacional de San Luis

19 de noviembre de 2013

Resumen

Las demandas actuales en el desarrollo del software implican una evolución y mantenimiento constante del mismo con el menor costo de tiempo y de recursos. Pensar en estrategias que faciliten las tediosas tareas que diariamente conllevan al crecimiento de los sistemas nos da incapie a iniciarnos en la investigación de herramientas automatizadas que reemplacen el esfuerzo manual que realizan los ingenieros de softwares a la hora de interpretar un programa. El correspondiente trabajo habla de técnicas basadas en el análisis de identificadores en códigos escritos en JAVA™

logrando así un aporte a nuestro principal objetivo que es comprender mejor los programas.

Capítulo 1

Introducción

1.1. Problema

Todos los problemas a los que se enfrentan los desarrolladores de software el primordial es el de mantener los sistemas en buen funcionamiento [4]. Esta tarea es imposible de llevar a cabo de forma manual debido a que consume muchos costos y esfuerzo humano. Por esta razón, existe una subárea de la Ingeniería de Software que se encuentra dedicada al desarrollo de técnicas de inspección y comprensión de software. Esta área tiene como principal objetivo que el desarrollador logre un entendimiento acabado del software de estudio de forma tal de poder modificarlo disminuyendo en lo posible la gran mayoría de costos [1]. El área mencionada se conoce en la jerga de la Ingeniería de Software como: *Comprensión de Programas (CP)*.

Uno de los principales desafíos en CP consiste en relacionar dos dominios muy importantes. El primero, el Dominio del Problema, hace referencia a la salida producida por el sistema de estudio. El segundo, el Dominio del Programa, se refiere a las componentes de software utilizadas para producir dicha salida.

Los caminos que conducen a facilitar la comprensión de software el mas apropiado consiste en el uso/creación de Herramientas de Comprensión. Una Herramienta de Compresión presenta diferentes perspectivas del software que posibilitan que el ingeniero pueda percibir el funcionamiento del sistema. Para construir herramientas de comprensión, se deben tener en cuenta tres pilares importantes, ellos son: *Interconexión de Dominios*, *Visualización de Software* y *Extracción de la Información* [3, 2].

La *visualización del software* es una característica importante en la comprensión de programas, básicamente provee una o varias representaciones visuales de algún sistema particular [1]. Dichas vistas, cuando están bien elaboradas, permiten analizar y percibir la información extraída desde un programa con mayor facilidad.

Por *Extracción de la Información* se entiende el uso/desarrollo de técnicas que permitan extraer información desde el sistema de estudio. Esta información puede ser: Estática o Dinámica, dependiendo de las necesidades del ingeniero de software o del equipo de trabajo.

Para la extracción de la información estática se utilizan técnicas de compilación tradicionales, que se encargan de recuperar información de cada componente del sistema. Todas las actividades que forman parte de esta tarea se realizan desde el código fuente sin ejecutar el sistema. Generalmente, en este tipo de trabajos se construye un analizador sintáctico con las acciones semánticas necesarias para extraer la información requerida. Por otro lado la extracción dinámica de información del sistema se obtiene aplicando técnicas de instrumentación de código, estas técnicas consisten en insertar sentencias dentro del código fuente del sistema con el fin de recuperar las partes del programa que se utilizaron para producir la salida. La principal diferencia que radica entre ambas técnicas es que las dinámicas requieren que el sistema se ejecute, mientras que las estáticas esto no es necesario.

Una de las vías mas sencillas de comprender grandes sistemas es relacionar el Dominio del Problema con el Dominio del Programa es por ello la necesidad de utilizar los conceptos basados en la *Interconexión de Dominios* [1]. Esta relación entre ambos dominios es compleja y se puede aproximar primero construyendo una representación de ambos dominios y luego llevar a cabo una estrategia de vinculación entre ambas representaciones.

1.2. Solución

El correspondiente trabajo

- Lograr una mejor comprensión de programas escritos en java a través del análisis de los identificadores encontrados en el código fuente de programas escritos con java.

Bibliografía

[1] Mario M. Berón, Daniel Riesco, and Germán Montejano. Estrategias para facilitar la comprensión de programas. San Luis, Argentina, April 2010.

[2] R. Brook. A theoretical analysis of the role of documentation in the comprehension of computer programs. *Proceedings of the 1982 conference on Human factors in computing systems*, pages 125–129, 1982.

[3] M. A. Storey, F. D. Fracchia, and H. A. Müller. Cognitive design elements to support the construction of a mental model during software exploration. *The Journal of Systems & Software*, 44(3):171–185, 1999.

[4] A. Von Mayrhauser and A. M. Vans. Program comprehension during software maintenance and evolution. *Computer*, 28(8):44–55, 1995.