Construcción de Representaciones del Dominio del Programa para Facilitar la Interconexión de Dominios

Ignacio Nadim El Kadre, Mario Berón, Carlos Salgado, Mario Peralta
Facultad de Ciencias Físico Matemáticas y Naturales - Universidad Nacional de San Luis
Ejército de los Andes 950 - San Luis- Argentina
email: ignacio.elkadre@gmail.com, {mberon,csalgado,mperalta}@unsl.edu.ar

Pedro Rangel Henriques
Departamento de Informática - Universidade do Minho
Braga-Portugal
email: pedrorangelhenriques@gmail.com

Maria J. Pereira
Departamento de Informática - Instituto Politécnico de Bragança
Bragança - Portugal
email: mjoao@ipb.pt

Resumen

La Comprensión de Programas es una disciplina de la Ingeniería de Software cuyo principal objetivo es elaborar métodos, técnicas y herramientas que ayuden al programador a entender programas.

Uno de los principales desafíos en esta disciplina consiste en relacionar dos dominios muy importantes como lo son: el dominio del problema y el dominio del programa. El primero hace referencia a la salida del sistema el segundo está relacionado con las componentes del programa utilizadas para producir esa salida.

La construcción de estrategias de interrelación de dominios implica el análisis y elaboración de técnicas de extracción de la información desde los sistemas. Dichas técnicas se clasifican de acuerdo a la clase de información que se extrae, en dinámicas y estáticas. Las primeras extraen información de tiempo de ejecución. Las segundas analizan el código fuente del sistema y muestran información relacionada con variables, constantes, etc.

El objetivo de la línea de investigación es desarrollar estrategias de extracción de información estática innovadoras que posibiliten: i) Navegar el código y ii) Ayuden a interconectar el dominio del problema con el dominio del programa.

Palabras clave: Comprensión de programas, Modelos, Ontologías, Mapeo conceptual.

1. Contexto

La línea de investigación descripta en este artículo se encuentra enmarcada en el contexto

del proyecto: *Ingeniería del Software: Conceptos Métodos Técnicas y Herramientas en un Contexto de Ingeniería de Software en Evolución* de la Universidad Nacional de San Luis. Dicho proyecto, es reconocido por el programa de incentivos y es la continuación de diferentes proyectos de investigación de gran éxito a nivel nacional e internacional.

También se forma parte del proyecto bilateral entre la Universidade do Minho (Portugal) y la Universidad Nacional de San Luis (Argentina) denominado *Quixote: Desarrollo de Modelos del Dominio del Problema para Inter-relacionar las Vistas Comportamental y Operacional de Sistemas de Software* [Quix11]. Quixote fue aprobado por el Ministerio de Ciencia, Tecnología e Innovación Productiva de la Nación (MinCyT) [MinCyT] y la Fundação para a Ciência e Tecnología (FCT) [FCT] de Portugal. Ambos entes soportan económicamente la realización de diferentes misiones de investigación desde Argentina a Portugal y viceversa.

2. Introducción

Para comprender un sistema de software, los responsables del mantenimiento a menudo deben invertir tiempo considerable en explorar y buscar artefactos de software, incluyendo el leer código fuente y/o documentos para localizar esas partes que son relevantes a una tarea de mantenimiento específica [YZ2007].

La Comprensión de Programas es una disciplina de la Ingeniería del Software cuyo objetivo es proveer Modelos, Métodos, Técnicas y Herramientas para facilitar el estudio y entendimiento de los sistemas de software.

Durante el proceso de comprensión de programas se abordan metodologías relacionadas a Visualización de Programas, Estrategias de Interrelación de Dominios y Métodos de Extracción de Información.

La Visualización de Software [BkK01a, Che06, Hen01] es una disciplina de la Ingeniería del Software cuyo objetivo es mapear ciertos aspectos de software en una o más representaciones multimediales [BRM2010]. Estas técnicas proveen vistas de alto nivel del sistema que incluyen grafos de llamadas, grafos de flujo de control, grafos de interacción entre clases, listas de posibles componentes o patrones candidatos, y lenguajes de descripción arquitectónica [LELP06].

Las Estrategias de Interrelación de Dominios contemplan la asociación entre el Dominio del Programa (la operación del programa) con el Dominio del Problema (el comportamiento del programa) [BRM2010]. Esto permite al programador relacionar objetos del programa con alguna funcionalidad específica facilitando el entendimiento del sistema y acortando tiempos en el mantenimiento del mismo.

Para la implementación de las técnicas de Visualización de Software y las Estrategias de Interrelación de Dominios se requiere la extracción de información del sistema que se está analizando, ya sea en funcionamiento (extracción dinámica) o partiendo del análisis del código fuente (extracción estática).

La extracción de información dinámica se centra en la recuperación de componentes usados en tiempo de ejecución. Una forma posible y comunmente usada consiste en la definición de un esquema de Instrumentación de Código. Este tipo de esquema consiste en la inserción de sentencias dentro del código fuente del sistema de estudio con la finalidad de recuperar las funciones y objetos de datos usados en tiempo de ejecución [BRM2010].

Para la extracción de información estática

se utilizan técnicas de compilación tradicionales donde se definen acciones semánticas apropiadas para extraer la información deseada.

La línea de investigación presentada en este artículo se centra en el estudio de estrategias de extracción de información estática útiles para interconectar el Dominio del Problema con el Dominio del Programa.

El artículo está organizado como sigue. La sección 2 describe sintéticamente la línea de investigación desarrollada en el proyecto. La sección 3 expone brevemente los resultados obtenidos/esperados en la temática abordada. Finalmente, la sección 4 menciona los trabajos planificados para la formación de recursos humanos.

3. Líneas de investigación y desarrollo

La comprensión de programas es esencialmente un proceso consistente en buscar, relacionar y recolectar información relevante sobre una pieza de software. Para facilitar esta tarea a los programadores, un enfoque exitoso debe considerar dos puntos fundamentales [YZ2007]:

1. ¿Qué tipo de información es la que busca el programador? y
2. ¿Cómo esas piezas de información están relacionadas?

El proceso de aprendizaje, necesario en Comprensión de Programas, tiene que ver con el proceso mental seguido por un programador cuando necesita entender un programa. Este tema es explicado en el contexto de *Modelos Cognitivos* [BHVU2007].

Un modelo mental de un programa de software consiste en relaciones entre elementos del código y el propósito e intención de esos elementos. Tal modelo mental puede tomar muchas formas, pero su contenido normalmente constituye una *ontología* [YZ2007].

En el contexto de la Comprensión de Programas, tal ontología consiste en conceptos y sus relaciones en el dominio de las técnicas de programación, estructuras de datos y algoritmos y técnicas de diseño. Para la construcción de esta ontología, llamada *ontología del programa*, se requiere la extracción de información del sistema.

Como ya se mencionó en la introducción de éste artículo, dicha extracción puede tener un enfoque dinámico, en el cual se recopila información del sistema en ejecución, y un enfoque estático a partir de la aplicación de técnicas de compilación del código fuente modificadas según el propósito del la tarea que se esté realizando.

Durante la secuencia normal de compilación, un programa es leído y traducido en otro lenguaje (el programa ejecutable). Este proceso de traducción se divide en dos etapas:

- El análisis propiamente dicho, en el cual el código fuente es separado en unidades (tokens) creando una representación intermedia del programa.
- La síntesis en la que se construye el programa ejecutable partiendo de la representación intermedia.

La etapa de análisis de código fuente consiste en 3 fases[ASU]:

- Análisis Lineal (Análisis Léxico), en el cual los caracteres del código fuente son leídos de izquierda a derecha y agrupados en *tokens*.
- Análisis Jerárquico (Análisis Sintáctico), en el cual los tokens son agrupados jerárquicamente en colecciones anidadas con sentido colectivo.

- Análisis Semántico, donde se ejecutan ciertos chequeos para asegurar que los componentes del programa encajan significativamente.

En todas las fases del proceso de compilación se van almacenando en una estructura todos los identificadores encontrados en el código fuente. Para cada uno de ellos se recuperan y se salvan en la misma estructura los atributos que determinan la función que éstos cumplen dentro del programa. Dicha estructura es conocida como *tabla de símbolos*. La información que ésta recolecta es utilizada durante el análisis y síntesis del programa y finalmente es descartada cuando la compilación se completa.

Este proyecto de investigación estudia estrategias de extracción de la información estáticas orientadas a simplificar la Comprensión de Programas. Pincipalmente se centra en la elaboración de estrategias que permiten construir la ontología del programa a partir de la recopilación de los distintos elementos almacenados en la tabla de símbolos y la relación existente entre ellos.

4. Resultados

Para construir una ontología del dominio del programa es necesario la implementación de un analizador sintáctico que permita la presitencia de la información presente en el código fuente. La tarea previamente mencionada puede ser llevada a cabo por medio de la construcción de una tabla de identificadores y la definición de un mecanismo de vinculación. Este mecanismo tiene como finalidad establecer una relación entre la información que se encuentra en la tabla de identificadores y una ontología del lenguaje de programación (Ésta ontología describe los conceptos propiamente dichos del lenguaje y las relaciones existentes entre ellos).

Para la construcción de la ontología mencionada en el párrafo precedente se utilizó *ANTLR* [ANTLR], que, como se describe en su página web, es una herramienta cuyo principal objetivo es la construcción de: reconocedores, intérpretes, compiladores y traductores a partir de descripciones gramaticales que contienen acciones en una variedad de lenguajes.

El lenguaje seleccionado para este proyecto es Java y para la generación de la tabla de identificadores se modificó la gramática del lenguaje disponible en el sitio web de *ANTLR*. Dicha modificación consitió en la inserción de reglas que posibilitan el almacenamiento de información sobre ciertos elementos del lenguaje en una base de datos (El motor de base de datos seleccionado fue **SQLITE** [SQLITE]).

La elección de una base de datos se fundamenta en el hecho de que éstas permiten almacenar de manera natural una tabla con elementos y otra tabla que permita relacionarlos.

Como trabajo futuro a corto plazo, se espera:

- Definir diferentes procedimientos que vinculen los identificadores recolectados durante los procesos de análisis sintáctico y semántico con los conceptos disponibles en la ontología del lenguaje de programación.

- Elaborar una estrategia de visualización que posibilite ver claramente: La tabla de Identificadores, La Ontología del Lenguaje de Programación y La Ontología del Dominio del Programa.

- Aplicar las estrategias a sistemas de gran envergadura.

5. Formación de Recursos Humanos

Las tareas realizadas en presente línea de investigación están siendo realizadas como parte del desarrollo de tesis para optar por el grado de Licenciado en Ciencias de la Computación. Se espera a corto plazo poder definir, a partir de los resultados obtenidos en las tesis de licenciatura en curso, tesis de maestría o bien de doctorado, como así también trabajos de Especialización en Ingeniería de Software. Es importante mencionar que tanto el equipo argentino como el portugués se encuentran dedicados a la captura de alumnos de grado y posgrado para la realización de estudios de investigación relacionados con las temáticas presentadas en este artículo. Dichos estudios pretenden fortalecer la relación entre la Universidad Nacional de San Luis y la Universidade do Minho.

Referencias

[BkK01a] Sarita Bassil and Rudolf k. Keller. A Qualitative and Quantitative Evaluation of Software Visualization Tools. Proc. of the IEEE Symposium on Information Visualization, pages 69–75, 2001.

[BRM2010] Mario M. Berón, Daniel Riesco, Germán Montejano, Pedro Rangel Henriques y Maria J. Pereira. Estrategias para Facilitar la Comprensión de Programas. Workshop de Investigadores en Ciencias de la Computación. El Calafate. Santa Cruz. Argentina. 2010.

[Che06] Chaomei Chen. Information Visualization. Springer; 2nd edition. ISBN: 1852337893. October 21, 2004.

[FCT] Fundação para a Ciência e a Tecnologia (FCT). Home Page. www.fct.mctes.pt/. 1997.

[Hen01] Gómez Henriques. Software Visualization: an Overview. Informatik, 2:4–7, 2001.

[LELP06] W. Lowe, M. Ericsson, J. Lundber, and T. Panas. Software Comprehension Integrating Program Analysis and Software Visualization. Technical Report, 2006.

[MinCyT] Ministerio de Ciencia, Tecnología e Innovación Productiva. Home page. http://www.mincyt.gov.ar/. 2007.

[Quix11] Página web del proyecto Quixote: http://www3.di.uminho.pt/ gepl/QUIXOTE/

[YZ2007] Yonggang Zhang. An Ontology-Based Program Comprehension Model. Ph.D. Thesis. Montreal. Quebec. Canada. 2007.

[ASU] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. Compilers Principles, Techniques and Tools. Addison Wesley; US ed edition. ISBN: 0201100886. January 1, 1986.

[BHVU2007] Mario Berón, Pedro Henriques, Maria J. Varanda y Roberto Uzal. Program Inspection to inter-connect the Operational and Behavioral Views for Programa Comprehension. First York Doctoral Symposium on Computing. York. England. 2007.

[ANTLR] ANTLR, ANother Tool for Language Recognition, http://www.antlr.org/

[SQLITE] SQL database engine, http://www.sqlite.org/