Título: Análisis de Identificadores para Abstraer conceptos del Dominio del Problema.

Autor: Javier Azcurra Marilungo.

Dirección:

- **Director:** Dr. Mario Marcelo Berón.
- **Co-Director:** Dr. Germán Antonio Montejano.

Objetivos: Los objetivos principales de este trabajo final de Licenciatura son:

- Extraer identificadores de programas escritos en lenguaje JAVA.
- Analizar los identificadores extraídos.
- Construir en JAVA una herramienta que implemente los ítems anteriores.
- Evidenciar las posibilidades de relacionar la salida de un programa con los elementos involucrados que producen dicha salida.

Antecedentes:

La Comprensión de Programas (CP) [7, 6, 5, 32] es un área de la Ingeniería de Software cuyo objetivo principal es desarrollar métodos, técnicas y herramientas que faciliten al programador el entendimiento de las funcionalidades de los sistemas de software. Una forma de alcanzar este objetivo consiste en relacionar el Domino del Problema, es decir la salida del sistema, con el dominio del programa, o sea con las partes del programa utilizadas para generar la salida del sistema [36, 27, 8]. La construcción de esta relación representa el principal desafío en el contexto de la CP.

Una solución posible al desafío previamente mencionado consiste en construir una representación para cada dominio y luego vincular ambas representaciones. La representación de ambos dominios se construye en base a la información, estática y dinámica, que se extrae de los mismos. En la actualidad se conocen distintas técnicas y herramientas que llevan a cabo esta tarea [1, 3].

La información dinámica se extrae del sistema en tiempo de ejecución. Las técnicas encargadas de extraer este tipo de información, insertan sentencias nuevas en el código fuente del programa. Esta inserción no altera la funcionalidad original del programa. La finalidad de las nuevas sentencias es registrar las actividades realizadas durante la ejecución del programa.

El código nuevo puede ser enriquecido con sentencias que simplemente impriman las secciones que se están ejecutando. Más complejo aún, cambiar un conjunto de

sentencias por uno nuevo para hacer un estudio más preciso en ciertas partes del código.

El volumen de información obtenida puede ser de gran magnitud. Es por esto, que el encargado de realizar esta tarea debe conocer con claridad los lugares estratégicos donde colocar las sentencias.

La instrumentación de código es una técnica de análisis dinámico. Esta técnica inserta sentencias para crear trazas de ejecución en distintas partes del código. Las trazas clarifican la ejecución del programa mejorando la comprensión del mismo.

Por otro lado, la información estática se extrae desde el código fuente del sistema usando técnicas de compilación tradicionales. Una de estas técnicas consiste en armar y recorrer el árbol de sintaxis abstracta (ASA). Esto permite realizar un análisis sintáctico y semántico en el código fuente.

Metodología:

Plan y Cronograma de tareas:

1. Estudiar las técnicas, herramientas y conceptos relacionados con la comprensión de programas: esto permite conocer todos aquellos aspectos vinculados con la comprensión de sistemas;

Recursos:

El correspondiente trabajo final se llevará a cabo en la Universidad Nacional de San Luis, en el Área Programación y Metodologías de Desarrollo de Software enmarcado en los siguientes proyectos:

- *"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.
- "Quixote - Development of Problem Domain Models to Interconnect the Behavioral and Operational Views to Aid in Software Systems Comprehension" (código de proyecto: PO/09/38). Quixote es un proyecto bilateral entre la Universidade do Minho (Portugal) y la Universidad Nacional de San Luis (Argentina). Dicho proyecto fue aprobado por el Ministerio de Ciencia, Tecnología e Innovación Productiva de la Nación (MinCyT)[1], y la Fundação para a Ciência e Tecnología (FCT)[2] de Portugal.

[1] www.mincyt.gov.ar/
[2] www.fct.mctes.pt/

Ambos entes soportan económicamente la realización de diferentes misiones de investigación desde Argentina a Portugal y viceversa, como así también la presentación y publicación de artículos en diferentes congresos nacionales e internacionales.

Por otro lado, el equipamiento necesario es una PC con el sistema Linux, impresora y papel para realizar las impresiones del informe. También se hará uso de internet, de material bibliográfico provisto por la biblioteca "Antonio Esteban Agüero"; y de material provisto por las librerías digitales a las que se tiene acceso desde la Universidad.

Bibliografía

[1] R. Sethi A. V. Aho and J. D. Ullman. Compilers principles, techniques and tools. 2006.

[2] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Estructuras de datos y algoritmos.* Sistemas Tecnicos de Edicion, 1988.

[3] Thoms Bell. The concept of dynamic analysis. 1999.

[4] Keith H. Bennett and Vaclav T. Rajlich. Software maintenance and evolution: a roadmap. 2000.

[5] M. Beron, P. Henriques, and R. Uzal. *Inspección de Programas para Interconectar las Vistas Comportamental y Operacional para la Comprensión de Programas.* PhD thesis, Universidade do Minho, Braga. Portugal, 2010.

[6] Mario Beron, Pedro R. Henriques, Maria J. Varanda, and Roberto Uzal. Program inspection to inter-connect the operational and behavioral views for program comprehension. 2007.

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

[8] 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.

[9] Bruno Caprile and Paolo Tonella. Nomen est omen: Analyzing the language of function identifiers. IEEE Computer Society, 1999.

[10] Bruno Caprile and Paolo Tonella. Restructuring program identifier names. IEEE Computer Society, 2000.

[11] Daniela da Cruz, Pedro Rangel Henriques, and Jorge Sousa Pinto. Code analysis: Past and present. *Proceeding of the Third International Workshop on Foundations and Techniques for Open Source Software Certification*, 2009.

[12] Dawn Lawrie Dave Binkley, Marcia Davis and Bonita Sharif. The impact of identifier style on effort and comprehension. *Empirical Software Engineering*, 2013.

[13] Henry Feild Dawn Lawrie and David Binkley. Quantifying identifier quality: an analysis of trends. Kluwer Academic Publishers-Plenum Publishers., 2006.

[14] Henry Feild Dawn Lawrie and David Binkley. Syntactic identifier conciseness and consistency. 2006.

[15] Henry Feild Dawn Lawrie and David Binkley. Extracting meaning from abbreviated identifier. 2007.

[16] Henry Feild Dawn Lawrie, Christopher Morrell and David Binkley. What's in a name? a study of identifiers. 2006.

[17] Thomas Eisenbarth, Rainer Koschke, and Daniel Simon. Aiding program comprehension by static and dynamic feature analysis. page 602, 2001.

[18] Ramez A. Elmasri and Shankrant B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1999.

[19] David W. Embley. Toward semantic understanding: an approach based on information extraction ontologies. Australian Computer Society, Inc. Darlinghurst, Australia, Australia, 2004.

[20] Haley Boyd Emily Hill, Zachary P. Fry. Automatically mining abbreviation expansions in programs to enhance software maintenance tools. 2008.

[21] Lori Pollock Eric Enslen, Emily Hill and K. Vijay-Shanker. Mining source code to automatically split identifiers for software analysis. 2009.

[22] Henry Feild, David Binkley, and Dawn Lawrie. An empirical comparison of techniques for extracting concept abbreviations from identifiers. In *In Proceedings of IASTED International Conference on Software Engineering and Applications (SEA 2006*, 2006.

[23] Henry Feild, David Binkley, and Dawn Lawrie. Identifier splitting: A study of two techniques. Abril 2006.

[24] Fangfang Feng and W. Bruce Croft. Probabilistic techniques for phrase extraction. *Inf. Process. Manage.*, 37:199–220, 2001.

[25] José Luís Freitas, Daniela da Cruz, and Pedro Rangel Henriques. The role of comments on program comprehension. 2008.

[26] Wilhelm Hasselbring, Andreas Fuhr, and Volker Riediger. First international workshop on model-driven software migration. Marzo 2011.

[27] Michael P. O'Brien. Software comprehension – a review and research direction. 2003.

[28] DeiBenbock F Pizka M. Concise and consistent naming. Institut fur Informatik, Technische Universitat Munchen Boltzmannstr, 2005.

[29] Roger S. Pressman. *Ingeniería del Software un enfoque práctico*. 2002.

[30] T.A. Standish. *Data structure techniques*. Computer Sciences. Addison-Wesley, 1980.

[31] 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.

[32] Margaret Anne Storey. Theories, methods and tools in program comprehension: Past, present and future. 2005.

[33] Pierre F. Tiako. Maintenance in joint software development. 2002.

[34] Tim Tiemens. Cognitive models of program comprehension. 1989.

[35] Marco Torchiano and Massimiliano Di Penta. Software migration projects in italian industry: Preliminary results from a state of the practice survey. 2007.

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

Firma de alumno y asesores