Análisis Dinámico para la Creación de Estrategias de Comprensión de Programas

Hernán Bernardis[(1)], Mario Berón[(1)], Daniel Riesco[(1)], Carlos Salgado[(1)], Pedro Rangel Henriques[(2)]
[(1)]Departamento de Informática/Facultad de Ciencias Físico-Matemáticas y Naturales/ Universidad Nacional de San Luis
Ejército de los Andes 950 – San Luis - Argentina
{hbernardis, mberon, driesco, csalgado}@unsl.edu.ar
[(2)] Departamento de Informática/Universidade do Minho
Braga – Portugal
pedrorangelhenriques@gmail.com

RESUMEN

La *Comprensión de Programas* (CP) es un área de la Ingeniería de Software que tiene como objetivo facilitar el entendimiento de los sistemas, mediante el desarrollo de Métodos, Técnicas, Estrategias y Herramientas que permiten comprender las funcionalidades del sistema de estudio.

Uno de los principales desafíos en CP es establecer una relación entre el *Dominio del Problema* y el *Dominio del Programa*. Es decir, poder relacionar el comportamiento del sistema de estudio con las componentes del mismo que producen dicho comportamiento. Una forma de construir esta relación consiste en elaborar una representación para cada dominio y luego establecer un procedimiento de vinculación entre ambas representaciones. Pero para lograr esto, es necesario poder extraer información de ambos dominios (para poder crear las representaciones), para lo cual existen múltiples técnicas.

Dentro de lo que a la extracción de información del programa se refiere, existen muchos métodos y herramientas desarrolladas, cada una de las cuales pueden ser clasificadas en base al tipo de información que extraen. Así, se tienen técnicas de extracción de información estática o dinámica. Las primeras extraen información desde el código fuente sin ejecutar el sistema. Las segundas están relacionadas con información de tiempo de ejecución.

En este artículo se describe una línea de investigación que se centra en el *Análisis Dinámico de Sistemas de Software* para la creación de estrategias de Comprensión de Programas. El Análisis Dinámico de Sistemas abarca el estudio de las técnicas de extracción de información junto a las técnicas y estrategias para la observación, estudio e interpretación de la información extraída.

Palabras Claves: Comprensión de Programas, Extracción de Información Dinámica, Análisis Dinámico, Programación Orientada Objetos.

CONTEXTO

La línea de investigación descripta en este artículo se desarrolla en el Laboratorio de Calidad e Ingeniería de Software de la Universidad Nacional de San Luis; y se encuentra enmarcada dentro del proyecto: *Ingeniería del Software: Conceptos, Métodos, Técnicas y Herramientas en un Contexto de Ingeniería de Software en Evolución*, perteneciente a la universidad antes mencionada. 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.

Este Trabajo también forma parte del proyecto bilateral entre la Universidade do Minho (Portugal) y la Universidad Nacional de San Luis (Argentina) denominado *Quixote*: *Development of Problem Domain Models to Interconnect the Behavioral and Operational Views to Aid in Software Systems Comprehension* [Quix11]. Quixote fue aprobado por el Ministerio de Ciencia, Tecnología e Innovación Productiva de la Nación (MinCyT) y la Fundação para a Ciência e Tecnología (FCT) de Portugal. Ambos entes

soportan económicamente la realización de diferentes misiones de investigación desde Argentina a Portugal y viceversa.

1. INTRODUCCIÓN

La Comprensión de Programas [Nel96, STO05, LF94, BRM10, CBHP08, Ber11] es un área de la Ingeniería del Software destinada a elaborar *Modelos*, *Métodos*, *Técnicas* y *Herramientas*, basadas en procesos de aprendizaje y en procesos específicos de ingeniería, con el fin de llegar a un conocimiento profundo sobre un sistema de software. Este aprendizaje es primordial para poder establecer una relación entre el Dominio del Problema y el Dominio del Programa. Es decir, lograr una relación real entre el comportamiento del sistema en estudio y las componentes del mismo utilizadas para producir dicho comportamiento. Para reproducir esta relación entre ambos dominios, se puede construir una representación para cada uno de ellos y luego, mediante una estrategia de vinculación, lograr conectar ambas representaciones; tal como se muestra en la Figura 1.

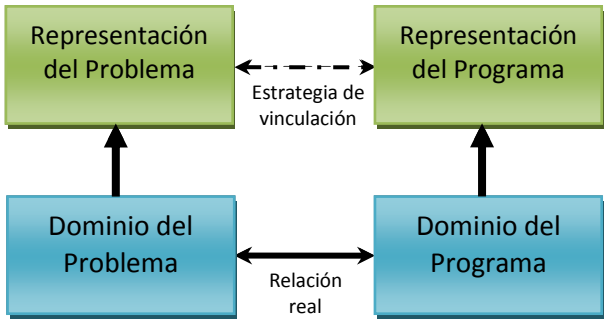

Figura 1: Modelo para la re-construccion de la relación real entre el Dominio del Problema y el Dominio del Programa.

Para poder construir las representaciones de los dominios, es necesario obtener información de ellos. Particularmente, en lo que a extracción de información desde el Dominio del Programa se refiere, surgen dos enfoques de investigación basados en el tipo de información que extraen, ellos son: el Análisis Estático (AE) y el Análisis Dinámico (AD). El AE es el más conocido debido a que es utilizado para la compilación de programas, y se centra en la extracción de información mediante el análisis sobre el código fuente del programa. El AD focaliza su trabajo sobre la información que se extrae durante el tiempo de ejecución del sistema, es decir, sobre el comportamiento que tiene el mismo mientras se ejecuta. Este último estilo de análisis provee una gran cantidad de ventajas, entre las que se pueden mencionar:

- Reducción del tamaño en el espacio de búsqueda respecto a las técnicas estáticas que analizan todo el código. Esto se debe a que las técnicas dinámicas extraen información de aquellas partes del programa involucradas en una ejecución particular. Un determinado escenario de ejecución[1] es muy difícil que abarque todo el código de un sistema.
- Detección de fallas producidas durante la ejecución que permiten descubrir y notificar errores que no se detectarían en un análisis estático, como por ejemplo el fallo de una operación con enteros.
- Extracción de información sobre el comportamiento del programa bajo distintos escenarios. Cada uno de estos escenarios produce un flujo de ejecución diferente dentro del programa, que emplea distintos componentes del mismo. Mediante el análisis dinámico se pueden registrar y analizar cada uno de esos flujos y las componentes del sistema usadas, de manera de poder establecer un vínculo entre estos que ayude a establecer la relación Dominio del Problema- Dominio del Programa.

Las ventajas mencionadas anteriormente son algunas de las razones por las cuales esta línea de investigación se centró en el Análisis Dinámico de Sistemas. A continuación se describen los aspectos relacionados a esta línea de investigación, detallando de manera más concreta las actividades que se están llevando adelante para extraer información dinámica del sistema de estudio.

La organización de este artículo se expone a continuación. La sección 2 describe la línea de

[1]Un escenario de ejecución es una secuencia de entradas de usuario que disparan acciones de un sistema que llevan a un resultado observable [EKS01].

investigación. La sección 3 presenta los resultados obtenidos hasta el momento, junto con aquellos esperados a corto plazo. Finalmente, la sección 4 describe las tareas realizadas por los recursos humanos en formación.

2. LÍNEA DE INVESTIGACIÓN

Como se mencionó anteriormente en el Resumen de este artículo, el Análisis Dinámico de Sistemas de Software está compuesto principalmente por dos amplias ramas de investigación, ellas son: i) las técnicas de extracción de información dinámica desde el programa, y ii) las técnicas y estrategias usadas para la administración, estudio e interpretación de la información extraída por las técnicas mencionadas en el ítem anterior.

Dentro de lo que a técnicas para la extracción de información dinámica desde los sistemas se refiere, se pueden mencionar 3 de las más conocidas y utilizadas actualmente como son:

- **Dynamic Program Slicing**: es una técnica que consiste en simplificar el análisis de los programas en ejecución focalizando el estudio sobre aquellas partes del código que poseen una mayor importancia semántica, descartando las restantes. Para lograr esto, se toman aquellas partes del código fuente de interés, y se las agrupa en nuevo/s procedimiento/s formando un nuevo sistema. Luego, al ejecutarse este nuevo sistema, se centra el análisis sobre el comportamiento de los nuevos procedimientos, obteniendo información de estos. Debido a que estos nuevos procedimientos no son más que partes aisladas del código fuente original, lo que se obtiene en realidad es información respecto a esa parte de código particular del sistema original [AH90].
- **Profiling**: esta técnica consiste en el uso de herramientas conocidas como *profilers* que se encargan de recolectar información sobre el comportamiento de los sistemas en ejecución. La información que estas herramientas proveen comprende desde el uso de memoria del programa hasta la duración de las llamadas a procedimientos. Para lograr obtener esta información, los *profilers* hacen uso de una amplia variedad de técnicas tales como interrupciones de hardware, simulación, uso de contadores de performance de hardware, entre otras [GKM82].
- **Instrumentación de Código**: esta técnica consiste básicamente en la inserción de sentencias de extracción de información dentro del código fuente de los programas, de manera que al ejecutarse se obtenga información respecto del comportamiento del sistema. Para más información leer [DGN07, BHU09, Ber11].

Todas estas técnicas dinámicas proveen grandes cantidades de información, las cuales deben ser analizadas y estudiadas para poder lograr llegar a una comprensión concreta de los programas. Por esta razón, el Análisis Dinámico se encarga de crear técnicas de manejo y estudio de la información para poder aprovecharla de la mejor manera. Entre las técnicas existentes se pueden mencionar:

- **Técnicas de Trace Summarization**: consisten en crear vistas de las trazas de ejecución más abstractas tomando los puntos principales de estudio de mayor interés y descartando la información referida a detalles. De esta forma, los ingenieros pueden crear un modelo cognitivo general de la traza, que les sirva de guía para luego realizar un análisis más detallado. El lector interesado en conocer más sobre esta técnica puede leer [HamL05],
- **Técnicas de Visualización de Trazas**: se utilizan para crear representaciones de las trazas más prácticas y gráficas que faciliten el análisis del contenido de las mismas. Así,

por ejemplo, se tienen técnicas de visualización de trazas basadas en árboles, grafos, diagramas, entre otras. El lector interesado en profundizar sobre estas técnicas puede leer [BHU09, Corn09, HamL05].

- **Técnicas de Interconexión de los Dominios del Problema y del Programa**: utilizan la información provista por las trazas de ejecución (Dominio del Programa) y la vinculan con información del Dominio del Problema para lograr establecer conexiones entre ambos dominios que permitan llegar a una comprensión más concreta de los sistemas [BHU09, LF94, BRM10].

En esta línea de investigación, basada en el Análisis Dinámico de los sistemas, se utilizan y elaboran técnicas tanto para la extracción de información como para el estudio administración e interpretación de la misma. Debido a la complejidad que conlleva desarrollar técnicas de extracción y de análisis de información, se decidió acotar la investigación sobre un paradigma de programación específico: el Orientado a Objetos, tomando como caso de estudio el lenguaje *Java*. La información que se extrae de los sistemas *Java* es referida al comportamiento en ejecución de los métodos del sistema, de manera de conocer qué métodos se ejecutan y en qué orden, para un determinado escenario de ejecución. Para extraer esta información de los sistemas se utiliza la técnica de Instrumentación de Código, aplicando diferentes acciones sobre la información extraída para poder aprovecharla de la mejor manera. Actualmente, se están analizando y definiendo más técnicas y estrategias para aplicar sobre la información que se extrae, de modo de realizar un análisis más completo sobre la misma, y llegar a una comprensión de los programas más acabada.

3. RESULTADOS OBTENIDOS Y ESPERADOS

Los resultados obtenidos hasta el momento en esta línea de investigación sobre Análisis Dinámico de Sistemas, son los siguientes:

- ✓ Se analizó una especificación de la gramática del lenguaje Java y se pudo concluir que es posible realizar una instrumentación mediante atributos sintetizados y heredados para la extracción dinámica de información [BBRHP11].
- ✓ Se definió un esquema de instrumentación que permite la extracción de información respecto del comportamiento de los métodos en ejecución y el control de cantidad extraída [Ber11].
- ✓ Se creó una herramienta que implementa el esquema de instrumentación definido, automatizando el proceso de instrumentación de los sistemas escritos en *Java* para obtener información dinámica de los mismos. Esta herramienta también administra la información extraída, permitiendo: i) visualización simultánea de la misma al momento de su extracción y, ii) registro en un archivo con formato XML para un análisis *post mortem* de la misma, mediante las diferentes herramientas existentes para XML, tales como creadores automáticos de árboles, grafos, entres otras.

Dentro de los resultados esperados en el corto plazo se encuentran:

- Crear ontologías sobre la información extraída de manera de facilitar la reconstrucción virtual de la relación real Dominio del Problema- Dominio del Programa. Se pretenden crear ontologías para describir los conceptos y las relaciones pertenecientes al Dominio del Problema, con el objetivo de sistematizar una representación de este dominio, y facilitar la identificación del mapeo de estos conceptos con secuencias de ejecución.
- Definir técnicas de análisis de trazas de ejecución para optimizar su estudio y comprensión.
- Definir estrategias de visualización de trazas de ejecución de manera de facilitar el entendimiento de la información extraída.
- Estudiar y crear técnicas de Trace Summarization para poder detectar, resaltar y centrar el análisis sobre la información extraída de mayor importancia.

- Definir nuevas estrategias de interconexión de dominios para la comprensión de programas.

Todas las tareas futuras mencionadas previamente permiten percibir la importancia de la línea de investigación presentada en este trabajo para la Comprensión de Programas y la Ingeniería Inversa.

4. FORMACIÓN DE RECURSOS HUMANOS

Las tareas realizadas en el contexto de la presente línea de investigación están siendo desarrolladas como parte de trabajos para optar al grado de Licenciado en Ciencias de la Computación. En el futuro se piensa generar diferentes tesis de maestría y doctorado a partir de los resultados obtenidos de los trabajos de licenciatura en curso.

5. REFERENCIAS

[Sto05] Storey, M. A. Theories, Methods and Tools in Program Comprehension: Past, Present and Future. XIII International Workshop on Program Comprehension (IWPC 2005). St Louis, USA. 2005.

[BRM10] Berón, M.; Riesco, D.; Montejano, G. Estrategias para Facilitar la Comprensión de Programas. XII Workshop de Investigadores en Cs de la Computación. El Calafate, Argentina. 2010.

[LF94] Lieberman, H.; Fry, C. Bridging the Gulf Between Code and Behavior in Programming. ACM Conference on Computers and Human Interface. Denver, USA. 1994.

[CBHP08] Cruz, D.; Berón, M.; Henriques, P.; Pereira, M. J. Strategies for Program Inspection and Visualization. International Scientific Conference on Computer Science and Engineering. Stará Lesná, Eslovaquia. 2008.

[DGN07] Denker, M.; Greevy, O.; Nierstrasz, O. Supporting Feature Analysis with Runtime Annotations. 3º International Workshop on Program Comprehension through Dynamic Analysis. Holanda. 2007.

[EKS01] Eisenbarth, T.; Koschke, R.; Simon, D. Aiding Program Comprehension by Static and Dynamic Feature Analisys. Proceedings of the International Conference on Software Maintenance ICSM'01. Florencia, Italia. 2001.

[Nel96] Nelson, M. A Survey of Reverse Engineering and Program Comprehension. Software Engineering Survey. 1996.

[BHU09] Berón, M.; Henriques, P.; Uzal, R. Inspección de Programas para Interconectar las Vistas Comportamental y Operacional para la Comprensión de Programas. Tesis doctoral de Universidad Nacional de San Luis (Argentina)-Universidade do Minho (Portugal). 2009.

[**HamL05**] Hamou-Lhadj, A. Techniques to Simplify the Analysis of Execution Traces for Program Comprehension; PhD Thesis, University of Ottawa. Canadá. 2005.

[**Corn09**] Cornelissen, S. Evaluating Dynamic Analysis Techniques for Program Comprehension. PhD Thesis, Universidad Técnica de Delft. Holanda. 2009.

[AH90] Agrawal, H; Horgan, J. Dynamic Program Slicing. Journal: SIGPLAN Notices. New York, USA. 1990.

[GKM82] Graham, S.; Kessler, P.; McKusick, M. K. Gprof: a Call Graph Execution Profiler. Journal: SIGPLAN Notices. University of California. USA.1982.

[BBRHP11] Bernardis, H.; Beron, M.; Riesco, D.; Henriques, P.; Pereira, M. J. Extracción de Información Dinámica en Programación Orientada a Objetos (Java). Proceedings de XIII Workshop de Investigadores en Ciencias de la Computación. Rosario, Argentina. 2011.

[Ber11] Bernardis, H. Instrumentación de Programas Escritos en Java para Interconectar los Dominios del Problema y del Programa. 40º Jornadas Argentinas de Informaticas JAIIO-EST. Córdoba, Argentina. 2011.

[Quix11] Proyecto Quixote Home Page http://www3.di.uminho.pt/~gepl/QUIXOTE/. 2011.