Visualización de Software: Conceptos, Métodos y Técnicas para Facilitar la Comprensión de Programas

Enrique A. Miranda, Mario Berón, Germán Montejano, Daniel Riesco
Departamento de Informática-Facultad de Ciencias Físico Matemáticas y Naturales
Universidad Nacional de San Luis
{eamiranda,mberon,gmonte,driesco}@unsl.edu.ar

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

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

Resumen

La Comprensión de Programas es una disciplina de la Ingeniería de Software cuyo principal objetivo es facilitar el entendimiento de los sistemas. Un aspecto importante involucrado en la Comprensión de Programas es la Visualización de Software (VS). La VS es una disciplina de la Ingeniería del Software que provee una o varias representaciones visuales de la información de los sistemas permitiendo una mejor compresión de los mismos. Dichas representaciones (también conocidas como vistas) no son fáciles de construir porque se deben tener en cuenta muchos factores cognitivos y de implementación. Los primeros son importantes porque sirven como puente cognitivo entre los conocientos que posee el programador y los conceptos usados en el sistema que se pretende comprender. Los segundos adquieren importancia porque la implementación de los puentes cognitivos es compleja y requiere de herramientas adecuadas para su concretización en una herramienta de comprensión.

Este artículo presenta una línea de investigación que estudia la Visualización de Software, una componente fundamental para la Comprensión de Programas. Dicha línea aborda principalmente el estudio y elaboración de: Estrategias de Visualización, Vistas y Herramientas de Visualización, entre otras tantas temáticas importantes referentes a Visualización de Software. Todas las temáticas mencionadas previamente son basales en Comprensión de Programas y son brevemente descriptas a lo largo de este artículo.

Palabras Claves: Visualización de Software, Comprensión de Programas, Librerías de Visualización de Software.

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 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 Interrelacionar las Vistas Comportamental y Operacional de Sistemas de Software* [Qui]. Quixote fue aprobado por el Ministerio de Ciencia, Tecnología e Innovación Productiva de la Nación (MinCyT) [Min07] 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.

1. Introducción

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 [BHU10, Wal02].

A través de un extenso estudio y experiencia en el desarrollo de productos de comprensión se pudo comprobar que el principal desafío en esta área consiste en relacionar el Domino del Problema con el Dominio del Programa. El primero hace referencia a la salida del sistema. El segundo a las componentes de software usadas para producir dicha salida. La figura 1 muestra un modelo de comprensión que sirve de base para reproducir la relación antes mencionada. El modelo declara que entre el Dominio del Problema y el Dominio del Programa existe una relación real que será reconstruida

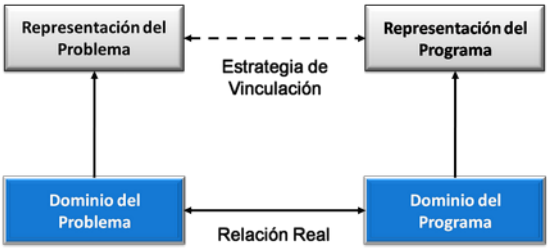

Figura 1: Modelo de Comprensión de Programas

a nivel virtual con la finalidad de facilitar la comprensión [BHU10, LF94].

La construcción de este tipo de relación es muy compleja e implica: i) Construir una representación para el Dominio del Problema; ii) Construir una representación del Dominio del Programa y iii) Elaborar un procedimiento de vinculación.

Si bien los tres pasos antes mencionados son de extrema importancia para elaborar "verdaderas" estrategias de comprensión, hace falta una componente de similar relavancia, esta es: La Visualización de Software [BHU10, SFM97].

2. Línea de Investigación: Visualización de Software

La Visualización de la Información posee el potencial de ayudar a las personas a encontrar la información que ellos necesitan, más eficientemente e intuitivamente [SL97, PQ06, Gó01, MB11]. La Visualización de Software (VS), tiene que ver con la representación visual de la información que proporcionan los sistemas de software. Esta representación se basa en su estructura, historia o comportamiento. La misma es una disciplina de la ingeniería del software que simplifica el análisis y la comprensión de los sistemas de software con el fin de mantener, re-usar, y aplicar re-ingeniería a los mismos. Hoy en día este aspecto se torna trascendental debido a que los sistemas de soft-

ware son cada vez más grandes y complejos tornando su desarrollo y mantenimiento más engorroso involucrando cada vez más la colaboración de muchas personas. Esto hace las tareas de programación, entendimiento y modificación del software más difícil, especialmente cuando se trabaja sobre el código de otra persona. Recientemente se han dado a conocer muchas herramientas y técnicas de visualización de software para dar apoyo a actividades como análisis, modelado, prueba, depuración y mantenimiento [BHU10]. Muchas herramientas y sistemas de software completos han sido desarrollados con el propósito de materializar las técnicas y enfoques investigados en el área de la visualización del software. A dichos sistemas se los denomina Sistemas de Visualización de Software (SVS). Los SVS proveen información del sistema analizado generando ciertas visualizaciones denominadas por muchos autores como "vistas". Una vista, visualización o artefacto visual es una representación de la información de un sistema que facilita la comprensión de un aspecto del mismo, es decir es una perspectiva del sistema. Las mismas actúan como puente coginitivo entre los conocimientos que posee el programador y los conceptos usados por el sistema. Existen distintos tipos de vistas dependiendo de la información que se desea visualizar. Cuando un programador está comprendiendo un programa, la primera vista con la que se encuentra es el código fuente del mismo. Esta vista es útil porque el programador está familiarizado con los lenguajes de programación. Sin embargo, cuando el tamaño del sistema crece, pierde claridad y otras perspectivas del sistema son necesarias. Teniendo en cuenta el modelo de Comprensión de Software desarrollado en la sección anterior, muchos sistemas de Visualización de Software presentan diferentes visualizaciones del programa (Domino del Programa) que son útiles para entender programas, pero no contemplan otras interesantes como es la de salida de sistema (Dominio de Problema) y su relación con los componentes del programa. Este problema dio orígen a un nuevo tipo de Sistemas Visualización de Software: Los Sistemas de Visualización de Software orientados a la Comprensión de Programas (PC-SVS) [BHU10]. Esta clase de sistemas (PC-SVS) tiene las mismas características que los de visualización de software tradicionales, con la diferencia que los nuevos deberían incorporar visualizaciones especiales orientadas a los Dominios del Problema y del Programa y la relación entre ellos [BHU10]. Los sistemas y herramientas de visualización de software han sido categorizados a través de los años de acuerdo a sus características y funcionalidades. Se han creado varias taxonomías para clasificar los sistemas y guiar a sus desarrolladores, a continuación se mencionan los más relevantes: Myers [BHU10], Price et. al. [BHU10], Roman and Kenneth [BHU10] y Storey et. al. [BHU10]. Los trabajos referenciados en el párrafo previo son los más influyentes en esta área de clasificación de Sistemas de Visualización de Software. Pero todos estos trabajos se refieren a la visualización del Dominio del Programa, dándole poca importancia a la visualización del Dominio del Problema. Recientemente, Berón y su grupo de investigación [BHU10] presentaron una extensión proporcionando una taxonomía para clasificar las Herramientas de Visualización de Software que tiene en cuenta el Dominio del Problema. Para concluir la sección cabe destacar que esta línea de investigación abarca el análisis de los PC-SVS, las distintas vistas que pueden generar y las diferentes estratégias de visualización que utilizan.

3. Resultados y Objetivos

La construcción de vistas requiere el estudio de diferentes librerías de visualización para representar fielmente los aspectos de software.

Cada una de estas librerías posee distintas características y atributos que las hacen diferentes entre sí, por lo cual la elección de una librería antes que otra es dependiente de la aplicación en donde se va a emplear y las distintas características de la misma. Para poder seleccionar la mejor librería para un caso de estudio específico, es necesario establecer un ranking entre las mismas. Teniendo en cuenta esta observación se escogió el método LSP (Logic Scoring of Preferences) [BHU10, OR02]. LSP es un método de comparación, evaluación y selección de distintos sistemas. El mismo se basa en la elección de distintos criterios que posibilitan a través de un proceso de agregación el establecimiento de un ranking. Para llevar a cabo esta tarea se precisa la definición de estos criterios y su posterior aplicación a las librerías de visualización. Esta tarea es muy importante ya que ayuda al diseñador en el complejo proceso de la elección de la librería de visualización más apropiada para una tarea específica. Si bien esta línea de investigación es de reciente creación, los resultados obtenidos hasta el momento, relacionados con los datos necesarios para el funcionamiento de LSP, constan de un cojunto de criterios que caracterizan a las librerías de visualización. Los mismos están clasificados en criterios generales que a su vez pueden contener subcriterios más específicos que permiten cuantificar el grado de satisfacción de los criterios más generales. En los ítems siguientes se describe la desagregación de los criterios más importantes, el lector interesado en tener una visión más profunda de los mismos puede leer [MB11].

- **Modelos Cognitivos:** Indica un conjunto de criterios que cubren los aspectos más importantes relacionados directamente con los procesos mentales usados por el ingeniero de software para la comprensión del sistema. Hasta el momento se han detectado los siguientes criterios: Interactividad, Sincronización e interacción entre vistas, Navegación, Evitar la sobrecarga de información, Presenta múltiples vistas que abarcan diferentes aspectos del sistema (Una descripción detallada de los mismos se puede en encontrar en [MB11]).
- **Vistas:** Cubre los aspectos más importantes relacionados con las vistas que se pueden generar. Este criterio se encuentra desagregado de la siguiente manera: i) Un conjunto de criterios orientados a la calidad de los artefactos que pueden presentar las vistas y ii) Un conjunto de criterios que describen los tipos de vistas que se pueden generar. Cada uno de esas desagregaciones están a su vez divididas en otros criterios de más bajo nivel. El lector intersado en inspeccionar los mismos puede leer [MB11].
- **Requerimientos:** Indica un conjunto de criterios que abarcan los requisitos básicos de hardware y software para las librerías de visualización. Por ejemplo: capacidad del procesador, capacidad de memoria, tipo de placa gráfica, etc. (ver [MB11] para obtener más detalles sobre los criterios mencionados anteriormente).

Como trabajo futuro a corto plazo se pretende: i) Completar el árbol de criterios obteniendo una desagregación completa y consisa de los aspectos más reelevantes de las librerías de visualización de software; ii) Estudiar, diseñar y construir diferentes vistas con el propósito de obtener nuevas o mejorar algunas existentes; y iii) Desarrollar un sistema que implemente LSP de forma genérica. Esto quiere decir que el usuario puede ingresar los criterios y definir sus propios mecanismos de agregación. iv) Utilizar el sistema mencionado en el ítem anterior para evaluar las librerías de visualización (utilizando los criterios definidos por el grupo de investigación).

4. Formación de Recursos Humanos

Las tareas realizadas en presente línea de investigación están siendo realizadas como parte del desarrollo de diferentes 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 antes mencionadas, tesis de maestría o bien de doctorado, como así también trabajos finales 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 temática presentada en este artículo. El objetivo principal de esta tarea es fortalecer la relación entre la Universidad Nacional de San Luis y la Universidade do Minho a través de la realización de sólidos trabajos de investigación.

Referencias

[BHU10] M. Beron, P. Henriques, and R. Uzal. Program Inspection to interconnect Behavioral and Operational Views for Program Comprehension. *Ph.D Thesis Dissertation. Universidade do Minho. Braga. Portugal*, 2010.

[FCT] Fundação para a ciência e a tecnologia (fct). www.fct.mctes.pt/. 1997.

[Gó01] H. Gómez. Software Visualization: an Overview. *Informatik*, 2:4–7, 2001.

[LF94] H. Lieberman and C. Fry. Bridging the gulf between code and behavior in programming. In *ACM Conference on Computers and Human Interface*, Denver, Colorado, April 1994.

[MB11] E. Miranda and M. Beron. Evaluación de funcionalidades de visualización de software provistas por librerías gráficas. Technical report, Universidad Nacional de San Luis, 2011.

[OR02] L. Olsina and G. Rossi. Measuring Web Application Quality with WebQEM. *IEEE MultiMedia, 2002*, 09(4):20–29, 2002.

[PQ06] M. Petre and E. Quincey. A Gentle Overview of Software Visualization. *PPIG: Psychology of Programing Interest Group*, pages 1–10, 2006.

[Qui] Quixote: Desarrollo de modelos del dominio del problema para interrelacionar las vistas comportamental y operacional de sistemas de software. http://www3.di.uminho.pt/ gepl. 2010.

[SFM97] M. A. Storey, F. D. Fracchia, and H. A. Muller. Cognitive design elements to support the construction of a mentalmodel during software visualization. *Program Comprehension, 1997. IWPC'97. Proceedings., Fifth Iternational Workshop on*, pages 17–28, 1997.

[SL97] J. Stasko and A. Lawrence. Empirically assessing algorithm animations as learning aids. *Software Visualization*, 1997.

[Wal02] A. Walenstein. Theory-based analysis of cognitive support in software comprehension tools. In *IWPC '02: Proceedings of the 10th International Workshop on Program Comprehension*, page 75, Washington, DC, USA, 2002. IEEE Computer Society.