Visualización de Software Orientada a Comprensión de Programas

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

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 proveer modelos, métodos, técnicas y herramientas para facilitar el entendimiento de los sistemas. Un aspecto importante 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 (también conocidas como vistas) de la información de los sistemas permitiendo una mejor compresión de los mismos. Dichas representaciones 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 conocimientos 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 aborda la Visualización de Software, una componente fundamental para la Comprensión de Programas. Dicha línea estudia: técnicas y estrategias de visualización, herramientas de visualización y creación de vistas. 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, Vistas.

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 Inter-relacionar las Vistas Comportamental y Operacional de Sistemas de Software* [4]. Quixote fue aprobado por el Ministerio de Ciencia, Tecnología e Innovación Productiva de la Nación (MinCyT) [9] y la Fundação para a Ciência e Tecnología (FCT) [12] de Portugal. Ambos entes soportan económicamente la realización de diferentes misiones de investigación desde Argentina a Portugal y viceversa.

1. Introducción

Sin lugar a dudas, una de las tareas más complejas y que más tiempo consume en el ciclo de vida de una aplicación es la de mantenimiento [14]. Esta etapa se basa en muchas actividades que necesitan

de la comprensión de programas, como por ejemplo: *Ingeniería Reversa*, *Reingeniería*, entre otras tantas disciplinas de la Ingeniería de Software (IS). La Comprensión de Programas (CP) es una disciplina de la IS cuyo objetivo es proveer modelos, métodos, técnicas y herramientas para facilitar el estudio y entendimiento de los sistemas de software [20, 2].

A través de un extenso estudio y análisis de herramientas de comprensión, se pudo comprobar que el principal desafío en esta área consiste en relacionar el Domino del Problema con el Dominio del Programa [2, 18]. 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. Dicho modelo declara que entre el Dominio del Problema y el Dominio del Programa existe una relación real que será reconstruida a nivel virtual con la finalidad de facilitar la comprensión [2, 8].

Figura 1: Modelo de Comprensión de Programas

La construcción de este tipo de relación es muy compleja e implica:

- Construir una representación para el Dominio del Problema.
- Construir una representación del Dominio del Programa.
- Elaborar un procedimiento de vinculación.

Si bien los tres pasos antes mencionados son de extrema importancia para elaborar "verdaderas" estrategias de comprensión, se debe tener en cuenta una componente de similar relevancia que permite visualizar las representaciones y el proceso de vinculación citado previamente. Dicha componente es: La Visualización de Software [2, 18].

2. Línea de Investigación: Visualización de Software

La VS es una disciplina de la Ingeniería de Software cuyo propósito es visualizar la información relacionada con los sistemas de software con el objetivo de simplificar el análisis y la comprensión de los mismos.

Muchas técnicas de visualización de software se utilizan para implementar Sistemas de Visualización de Software (SVS) [1, 18]. Un SVS es una herramienta que provee información del sistema analizado generando ciertas visualizaciones denominadas por muchos autores como "vistas". Una vista o visualización 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 cognitivo 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.

Teniendo en cuenta los tres pasos presentados en la sección anterior, muchos SVS exhiben diferentes visualizaciones del programa (Domino del Programa) que son útiles 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 origen a un nuevo tipo de Visualización de Software: Los Sistemas de Visualización de Software Orientados a la Comprensión de Programas (SVS-PC o PC-SVS) [2]. Esta clase de sistemas tiene las mismas características que los de visualización de software tradicionales, con la diferencia que los nuevos deberían incorporar vistas especiales orientadas a los Dominios del Problema y del Programa y la relación entre ellos [2, 3].

Para concluir esta sección cabe destacar que esta línea de investigación abarca el análisis de los PC-SVS, las distintas vistas que pueden generar, las diferentes estrategias de visualización que utilizan, las herramientas para la construcción de dichas vistas y demás aspectos relacionados a la VS.

3. Resultados y Objetivos

A continuación se mencionan los resultados obtenidos hasta el momento dentro del marco de la línea de investigación:

a) Se encontró que existen numerosos SVS, pero no presentan vistas orientadas al Dominio del Problema y a la relación de este con el Dominio del Programa. En pocas palabras, no existen en la actualidad numerosos PC-SVS. Por lo tanto, el trabajo que se lleva a cabo en la línea de investigación en conjunto con el proyecto Quixote [4] toma relevancia desde este punto de vista.

b) Se puede concluir que en la mayoría de los casos, las estratégias utilizadas por los PC-SVS son estáticas. Por lo que sería de gran utilidad incorporar estratégias de visualización dinámicas a las técnicas usadas por los sistemas actuales como propone Berón et. al. en [2]

c) Se confimó que una clase de herramienta a tener en cuenta a la hora de construir vistas son las librerías de visualización de software.

Como se mencionó en el párrafo precedente, la construcción de vistas requiere el estudio de diferentes librerías de visualización para representar fielmente los aspectos de software [7, 15]. Una librería de visualización de software es un conjunto de subprogramas que proveen distintas facilidades para el desarrollo de vistas. 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.

Este proceso de selección de la librería más conveniente, es llevado a cabo por el programador de manera intuitiva, tomando como posibles indicadores: la experiencia de programación en el área, recomendaciones de otros pares, librería utilizada en el ambiente donde se trabaja, etc. Decir que tales indicadores no son de utilidad, sería una falta de consideración al proceso que se ha llevando a cabo durante mucho tiempo. Sin embargo, no proveen al programador un procedimiento sistemático a la hora de la elección de la librería más adecuada para un caso particular.

Una manera de poder seleccionar la librería de visualización más adecuada para cada caso, consiste en definir criterios de comparación. Para ello, debería tenerse en cuenta algunas propiedades de los SVS, ya que muchas de estas son significativas al momento de analizar las librerías de visualización. Entre los trabajos relacionados con tales propiedades están aquellos afines a las clasificaciones de los sistemas de visualización de software [11, 16, 17].

Los trabajos referenciados en el párrafo anterior son los más influyentes en esta área de clasificación de los SVS. Pero todos ellos se refieren a la visualización del Dominio del Programa, dándole poca importancia a la visualización del Dominio del Problema. Recientemente, los integrantes del proyecto PCVIA [3, 13] presentan una extensión a la taxonomía de Price (Considerada la más completa por la jerga de visualización de software), proporcionando otra orientada a los PC-SVS.

Con el propósito de utilizar un mecanismo de evaluación sofisticado y a la vez flexible para la evaluación de las librerías de visualización, se procedió al estudio de diferentes métodos de evaluación. El resultado de dicho estudio consistió en la selección del método de evaluación Logic Scoring of Preference (LSP) [5, 6]. LSP es un mecanismo de comparación, evaluación y selección de distintos sistemas. Recibe como entrada un conjunto de características y atributos que el usuario pretende que el sistema posea. Luego por medio de un proceso de agregación, dichos atributos son evaluados y se elabora un ranking de los sistemas de acuerdo al resultado obtenido por el método.

Si bien la línea de investigación no posee mucho tiempo de creación, los resultados obtenidos hasta el momento, relacionados con los datos necesarios para el funcionamiento de LSP, constan de un conjunto de criterios que caracterizan a las librerías de visualización de software. Estos están clasificados en criterios generales, que a su vez pueden contener subcriterios más específicos los cuales permiten cuantificar el grado de satisfacción de los criterios más generales. El lector interesado en conocer cuáles son y que representan esos criterios puede estudiar [6, 10].

A su vez, se desarrolló una aplicación para la evaluación de distintos tipos de sistemas, basada en el método LSP. Para el desarrollo de la misma se utilizó, como parte del conjunto de herramientas de desarrollo, la librería gráfica Prefuse [15], intentando con esto obtener experiencia con las librerías que luego serán sometidas a evaluación.

La aplicación consta de cuatro etapas descriptas a continuación:

1. Definición Árbol de Criterios: se proveen facilidades para la construcción de un árbol de criterios correspondiente al tipo de sistema a evaluar (Figura 2).

2. Diseño Estructura de Agregación: permite

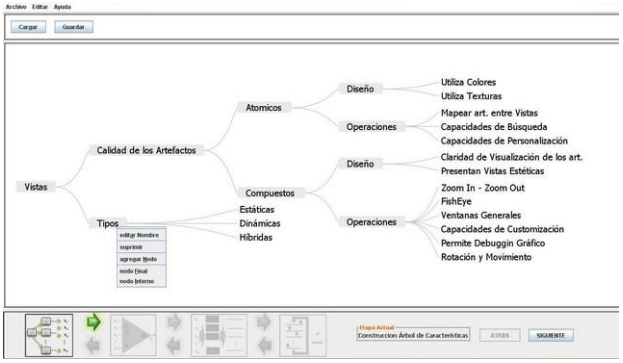

Figura 2: Aplicación - Etapa de Construcción

crear la estructura de agregación del conjunto de criterios definidos en la etapa anterior (nodos hojas del árbol) (Figura 3).

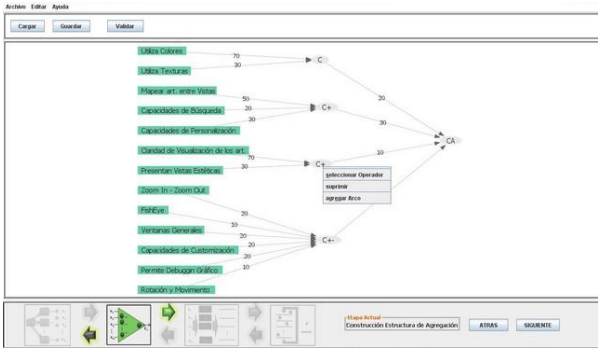

Figura 3: Aplicación - Etapa de Agregación

3. Definición Funciones de Criterios: se brindan facilidades para la definición de las funciones elementales correspondientes a cada criterio ([5]) (Figura 4).

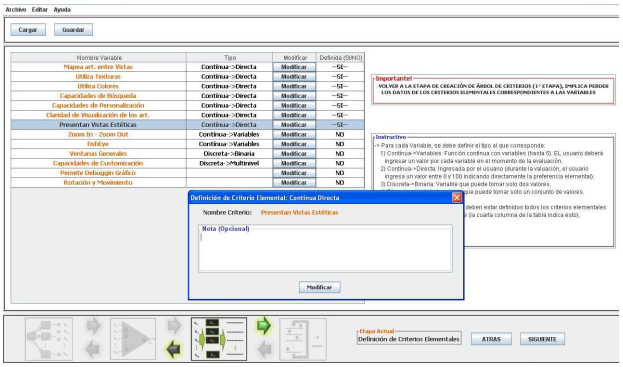

Figura 4: Aplicación - Etapa de Definición de Criterios

4. Evaluación: etapa final donde se evaluan los sistemas teniendo en cuenta todo lo definido en los pasos anteriores (Figura 5).

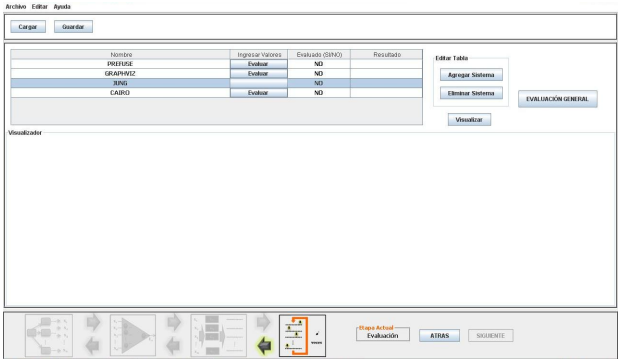

Figura 5: Aplicación - Etapa de Evaluación

La herramienta devuelve como resultado un ranking de los Sistemas a evaluar, dependiendo de los datos introducidos en cada etapa.

Como trabajo futuro a corto plazo se pretende:

i) A partir de un análisis profundo sobre el árbol de criterios propuesto [6], se encontró que la mayoría de los atributos que lo componen, caracterizan más a los SVS que a las librerías de visualización. Por lo tanto, se pretende depurar el conjunto de criterios propuesto e incorporar un grupo de criterios más específicos de las librerías de visualización de software. Por último, se busca estandarizar el árbol de requerimientos usando un estándar internacional.

ii) Evaluar distintas librerías de visualización y poner los resultados obtenidos a disposición de la comunidad de Ingeniería de Software. Para esto se utilizará la aplicación desarrollada y el árbol de criterios mencionado anteriormente.

iii) Estudiar, diseñar y construir diferentes vistas con el propósito de obtener nuevas perspectivas del software o mejorar algunas existentes.

iv) Seguir enfocando la línea de investigación en los PC-SVS y más precisamente en la utilización de animación y técnicas de visualización dinámicas ([19]), procurando enriquecer algunos sistemas ya existentes que no poseen estas características.

4. Formación de Recursos Humanos

Las tareas realizadas en la presente línea de investigación están siendo desarrolladas como parte de diferentes tesis de Licenciatura en Ciencias de la Computación en la Universidad Nacional de San Luis. Dichas tesis son supervisadas por docentes de

universidad de San Luis y docentes de la Universidade do Minho. Se pretende que los resultados obtenidos durante el desarrollo de las tesis den origen a estudios de posgrado en América del Sur o en Europa. Permitiendo, de esta manera, tanto el crecimiento académico de los integrantes de esta línea de investigación, como así también la continuación de trabajos de investigación conjuntos entre la Universidad de San Luis y la Universidade do Minho.

Referencias

[1] T. Ball and SG Eick. Software visualization in the large. *Computer*, 29(4):33–43, 1996.

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

[3] M. M. Beron, D. Cruz, M. J. Varanda Pereira, P. R. Henriques, and R. Uzal. Evaluation criteria of software visualization system used for program comprehension. *3a Conferencia Nacional em Interacção Pessoa-Máquina*, 03:285, 2008.

[4] 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.

[5] J.J. Dujmovic. A Method for Evaluation and Selection of Complex Hardware and Software Systems. *The 22nd Int'l Conference for the Resource Management and Performance Evaluation of Enterprise CS. CMG 96 Proceedings*, 1:368–378, 1996.

[6] Miranda Enrique. Evaluación de Funcionalidades de Visualización de Software Provistas por Librerías Gráficas. *40 Jornadas Argentinas de Informática e Investigación Operativa - EST. Cordoba Argentina.*

[7] GraphViz-Team. http://www.graphviz.org/, 2011.

[8] 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.

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

[10] Montejano Germán Riesco Daniel Henriques Pedro Pereira Maria J. Miranda Enrique, Berón Mario. Visualización de Software: Conceptos, Métodos y Técnicas para Facilitar la Comprensión de Programas. *XIII Workshop de Investigadores en Ciencias de la Computación. Rosario. Santa Fe. Argentina.*

[11] B. Myers. Taxonomies of Visual Programming and Program Visualization. *Journal of Visual Languages and Computing*, 1(1):97–123, 1990.

[12] Fundação para a Ciência e a Tecnologia (fct). http://www.fct.mctes.pt/, 1997.

[13] PCVIA-Team. http://wiki.di.uminho.pt /twiki/bin/view/research/pcvia/, 2011.

[14] M. Petrenko, V. Rajlich, and Vanciu R. Partial Domain Comprehension in Software Evolution and Maintenance. *International Conference on Program Comprehension (ICPC08).*

[15] Prefuse-Team. http://prefuse.org/, 2011.

[16] B. A. Price, I. S. Small, and R. M. Baecker. A taxonomy of software visualization. *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*, 2, 1992.

[17] G. C. Roman and K. C. Cox. A taxonomy of program visualization systems. *Computer*, 26(12):11–24, 1993.

[18] M. A. Storey. Theories, methods and tools in program comprehension: past, present and future. *Proceedings of the 13th International Workshop on Program Comprehension*, pages 181–191, 2005.

[19] M. J. Varanda. *Sistematização da Animação de Programas.* PhD thesis, Universidade do Minho, 2003.

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