APLICACIÓN DE LA VISUALIZACIÓN DINÁMICA DE PROGRAMAS EN UN MODELO DE ENSEÑANZA A DISTANCIA PARA INGENIEROS INFORMÁTICOS

Eje Temático: Recursos para el aprendizaje y la investigación de calidad.

M. Sc. Yolanda Soler Pellicer [(1)]; PhD. Mateo Gerónimo Lezcano Brito [(2)]; M. Sc. Isvani Frías Blanco [(1)]; Ing. Edel Ángel Rodríguez Sánchez [(1)]; Ing. Manuel Alarcón Suárez[(1)]; M. Sc. Manuel José Linares Alvaro [(1)]

[(1)] Universidad de Granma, Cuba; [(2)] Universidad Central "Marta Abreu" de Las Villas, Cuba

yoly@udg.co.cu, mlezcano@uclv.edu.cu, ifriasb@udg.co.cu, edel@udg.co.cu, malracons@udg.co.cu, cheche@udg.co.cu

Resumen: Las Técnicas de Visualización de Programas (TVP) realizan representaciones gráficas y abstracciones de alto nivel que describen el código y los datos del programa, transformando la información tradicional en una más significativa. En esta investigación se demuestra que la integración de varias TVP contribuye a solucionar problemas relacionados con el diseño e implementación de estructuras de datos y programas. Se propone el Ambiente Integrado de Visualización de Estructuras de Datos (VIA-ED) basado en mapas conceptuales que constituye un repositorio de recursos, uno de los cuales es el sistema VisualProg que tiene como entrada el código en el lenguaje SubC, desarrollado para este fin, y que integra los componentes de visualización de código, datos y complejidad del programa. VisualProg fue implementado teniendo en cuenta la arquitectura propuesta para el diseño de Sistemas de Visualización de Programas (Arq_VP), concebida en tres capas: Analizador de Código, Controladora y Vista. El ambiente se aplica en el proceso de enseñanza-aprendizaje de la asignatura Estructura de Datos de la carrera Ingeniería Informática en la Universidad de Granma y evaluación por los estudiantes y expertos. Se presenta un modelo de interacción de cada

componente del ambiente para un modelo de enseñanza a distancia.

Palabras Claves: Visualización de Programas, Estructuras de Datos, Mapas Conceptuales, Enseñanza a Distancia.

1. INTRODUCCIÓN

El aprendizaje de las disciplinas de Algorítmica y Programación de Algoritmos presenta, quizás, uno de los niveles de dificultad más elevado en las carreras de Ingeniería Informática y Ciencia de la Computación. Históricamente, los estudiantes han presentado problemas para asimilar nociones matemáticas abstractas, en particular cuando éstas incluyen la dinámica de cómo los algoritmos manipulan los datos (ACM y IEEE-CS, 2005).

Reconociendo la afirmación anterior, muchos centros de enseñanza en todo el mundo han dedicado grandes esfuerzos a resolver esta situación para lo cual han usado diferentes vías y herramientas que ayudan a mejorar el aprendizaje y el desarrollo de algoritmos, pero la problemática continúa vigente, lo que ha motivado el surgimiento de programas especiales que se usan para ayudar a explicar la conducta de otros programas. Una de las técnicas más usadas por estos programas son las de visualización. La Visualización de Información, en general, consiste en el uso de recursos gráficos, de animación y multimediales, con importante interacción entre el usuario y la computadora. Específicamente, la Visualización de Software usa recursos similares a la visualización de información y tiene como finalidad facilitar la comprensión y el uso efectivo de programas (Clinton, 2004; TechnM, 2005; Jeffery, 2006; Señas y Moroni, 2006). Diversos autores, entre ellos (Brown et al., 2005; Grinstein y Levkowits, 2005), consideran que las visualizaciones no son una percepción sensorial de un hecho sino una construcción mental que se acerca al conocimiento mediante la conciencia de un objeto real pero ausente o inexistente. En correspondencia, los datos se han venido transformando en información durante toda la historia de la humanidad, aún antes de que existieran las computadoras; pero el surgimiento de estas ha creado un paradigma desde diferentes puntos de vista que ha causado una revolución tecnológica en los últimos años en cuanto a la visualización se refiere.

En relación con la enseñanza de la programación se destaca la importancia de las actividades comprendidas en la construcción y la lectura comprensiva de programas. Se coincide con (Gugerty y Olson, 2006), al considerar un enfoque que parta de la resolución de problemas, para el primer aspecto, y que se apoye en técnicas de visualización de algoritmos y programas para el segundo caso. La Visualización de Software comprende la Visualización de Algoritmos y la de Programas. La primera consiste en la visualización de abstracciones de alto nivel que describen el algoritmo, mientras que la segunda se refiere al código real de programa y a las estructuras de datos. Ambas pueden darse en forma estática o dinámica (Frías et al., 2007).

Múltiples sistemas citados por (Moshell et al., 1987; Graf, 2007; Chang et al., 2008; Maimone et al., 2008), usan los gráficos para mostrar algunos aspectos del programa. Según (Myers, 2004; Naps y Rößling, 2004; Price et al., 2004; Satratzemi et al., 2004; Brown et al., 2005; Grinstein y Levkowits, 2005; Hundhausen, 2006; Stasko et al., 2006), existen otros Sistemas de Visualización de Programas que intentan ilustrar el código, los

datos o algoritmos de un programa ajustándose a múltiples categorías porque ilustran varios aspectos o muestran diferentes modos de visualización y manejo.

Generalmente, para lograr una interpretación acabada de un programa se siguen dos vías, la primera realiza el estudio del código fuente que es incómodo y de difícil aplicación y la segunda construye casos de prueba para explicar la conducta de un programa, tarea que resulta penosa y especulativa; ambas vías se pueden usar independientemente o de forma combinada. Estas dificultades motivan el desarrollo de programas especiales que se usan para ayudar a explicar la conducta de otros programas (Jeffery, 2006; Moroni y Señas, 2006).

Si bien los entornos de los lenguajes de programación ofrecen capacidades cada vez más útiles al programador, como el uso de colores y los depuradores, éstos no son aún lo suficientemente amigables con el usuario como es deseable. Se ha trabajado en el desarrollo de entornos específicos que ayuden y conduzcan efectivamente a los estudiantes en el proceso de aprendizaje de la programación de manera más gráfica, diversa y comprensible. (Naps y Rößling, 2004), consideran que la visualización de la conducta dinámica de los programas y de sus modelos abstractos, los algoritmos y, consecuentemente, el beneficio psicopedagógico de su aplicación en la enseñanza, constituyen una importante área de investigación que aún no ha sido completamente explorada.

Por otro lado, se deben tener en cuenta las necesidades de los programadores noveles. La teoría cognoscitiva de (Newell, 1972), seguida por (Anderson, 1983; Kearns y Vazirani, 2004; Brown et al., 2005; Grinstein y Levkowits, 2005; Deek et al., 2008) plantea, como principio fundamental, que la forma en que el conocimiento, las habilidades y las actitudes se aprenden inicialmente, juega un papel determinante sobre el grado en que dichas cualidades pueden usarse en otro contexto.

El primer tema de la asignatura Estructura de Datos de la carrera Informática está relacionado con las estructuras lineales, comenzando el estudio de las listas enlazadas. Se realiza una analogía con una estructura lineal ya estudiada, los arreglos, y se muestra la diferencia entre la representación de memoria secuencial y enlazada. Desde este momento se evidencian grandes insuficiencias en la definición de arreglos, por lo que se necesita profundizar en el manejo de la representación secuencial, dada la incidencia que tiene en el logro de los objetivos de la asignatura, lo que afecta directamente en la adecuada selección de las estructuras apropiadas para representar la información.

Al comparar los contenidos de las asignaturas de la disciplina con el sistema de habilidades que deben alcanzar los estudiantes en cada una, se revela que desde la primera asignatura de programación se exige eficacia en el diseño y elaboración de programas que hagan un uso eficiente de los recursos de un sistema de cómputo. Sin embargo, en el currículo de la carrera de Ingeniería Informática no existe una asignatura encargada de los contenidos relacionados con las técnicas de diseño y análisis de algoritmos. Estos temas se han distribuido entre las asignaturas de la disciplina de Ingeniería y Gestión de Software.

Todo ello constituye una problemática a la cual aún la ciencia no ha dado respuestas definitivas, por lo que se formula el siguiente **problema de investigación**:

Las Técnicas de Programación de Computadoras son difíciles de asimilar y los

métodos de enseñanza actuales no han resuelto totalmente el problema de seleccionar y diseñar las estructuras de datos adecuadas para representar la información de problemas complejos.

Esta situación problémica es una oportunidad que permite cambiar viejos paradigmas de enseñanza tradicional para elaborar un sistema de ayuda al diseño, en base a la experiencia acumulada y a las facilidades que brindan actualmente las Tecnologías de la Información y las Comunicaciones y la Informática Educativa, de ahí que, para transformar el objeto de investigación y apoyar la solución del problema científico, se propone el siguiente **objetivo general**:

Elaborar un ambiente de visualización de programas y datos compuesto por un conjunto de recursos integrados, que permita mejorar el diseño de estructuras de datos y programas y facilite análisis de eficiencia al mostrar, de forma dinámica, el comportamiento del código y los datos.

Las tareas derivadas de este objetivo fueron primeramente el resultado de un ensayo investigativo, validado en el proceso de enseñanza-aprendizaje de la asignatura Estructura de Datos de la carrera Ingeniería Informática en la Universidad de Granma, y en estos momentos se ha extendido como experiencia institucional en la Universidad Central “Marta Abreu” de Las Villas, la Universidad de Camaguey, la Universidad de Ciencias Informáticas de La Habana, el Centro Universitario de Las Tunas, la Universidad de Holguín y constituye uno de los productos académicos de la Red Nacional del Ministerio de Educación de Cuba. El acceso a VIA-ED desde cualquier universidad de Cuba y el mundo y los recursos que contiene han contribuido a su generalización.

2. DESARROLLO

Para dar cumplimiento al objetivo trazado se desarrolla el Ambiente de Visualización de Estructuras de datos que constituye a su vez un repositorio de recursos que integran múltiples TVP (Figura 1).

2.1. Ambiente Integrado de Visualización de Estructuras de Datos (VIA-ED)

En el análisis realizado por (Soler y Lezcano, 2007), se detecta, como uno de los problemas en el proceso de aprendizaje de las técnicas de programación de computadoras, el nivel de abstracción que tienen los conceptos objetos de estudio; la complejidad que puede alcanzar el diseño de las estructuras de datos, en dependencia del problema a resolver; la integración de las diferentes estructuras estudiadas en la solución de un problema; el logro de algoritmos eficientes y las escasas habilidades logradas en las asignaturas que le preceden.

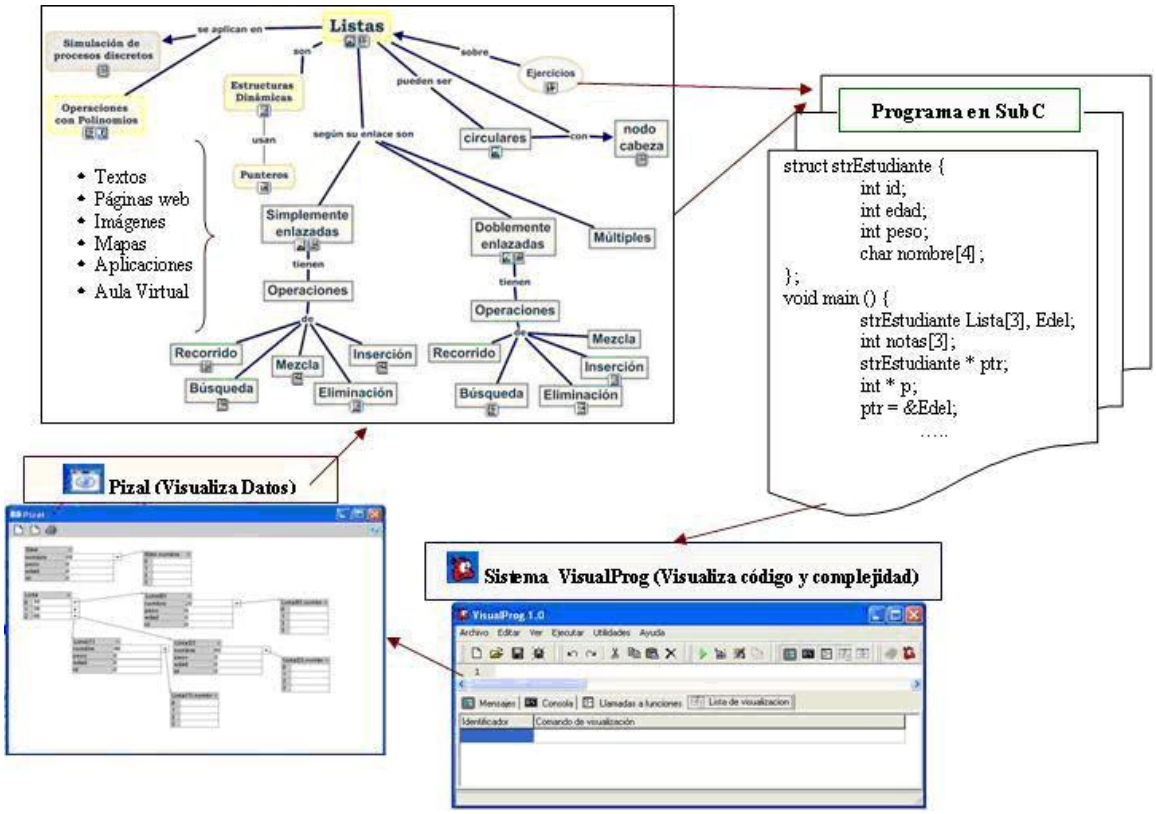

Figura 1. Esquema del Ambiente Integrado de Visualización de Estructuras de Datos

Teniendo presente la caracterización de las TVP y las deficiencias encontradas, se propone un ambiente basado en mapas conceptuales, que organiza e integra el modelo del conocimiento de la asignatura Estructura de Datos y sirve de interfaz visual y repositorio de recursos informáticos: simulaciones, animaciones, imágenes, útiles para el autoaprendizaje en un modelo de estudios semipresencial.

Usando las facilidades de la herramienta CmapTools instalada en la Universidad de Granma y con acceso a la Red Nacional de Computación del Ministerio de Educación Superior se diseñó el mapa conceptual Tipos Abstractos de Datos, integrado por otros 24 mapas conceptuales y 204 recursos, que incluyen documentos, imágenes, aplicaciones, simulaciones, mapas y el Sistema de Visualización de Programas en el lenguaje SubC.

El nuevo servidor de mapas conceptuales de la Universidad de Granma (http://cmap.udg.co.cu), actúa como un repositorio compartido de modelos de conocimientos que contiene los mapas, así como un índice para búsquedas de otros recursos insertados. El repositorio facilita la colaboración en la edición y construcción de mapas conceptuales usando hilos de discusión sincrónicos.

2.1.1. Informaciones textuales e imágenes

Las informaciones en diferentes formatos insertadas a VIA-ED recogen la teoría, métodos y modelos matemáticos relacionados con las estructuras de datos, los

algoritmos y programas que las implementan. Para cada programa propuesto se realiza un análisis de la complejidad, si son operaciones que puedan implementarse por varios métodos, como el caso del ordenamiento en arreglos (tema II de la asignatura Estructuras de Datos), se comparan y se hacen sugerencias sobre la factibilidad de usar uno u otro en dependencia del problema, la cantidad de datos a procesar y las operaciones.

2.1.2. Aplicaciones y simulaciones

A las operaciones con las estructuras de datos se le adicionan, como recursos del mapa, diferentes algoritmos o métodos que usan la Programación Visual para mostrar los objetos a la vez que se simulan sus transformaciones y el movimiento de índices o punteros. Se simulan, además, algoritmos complejos como el de Kruskal y Prim para construir árboles de cubrimiento de costo mínimo para grafos no dirigidos, conexos y con costos sobre las aristas y el de Dijkstra para resolver problemas del camino mínimo.

En la presente investigación se consideró que el recurso más importante del ambiente debía facilitar la visualización dinámica de datos y código de programas, mostrar las operaciones, permitir la realización de cambios en el programa y comprobar el efecto que provoca en los datos. Por lo que se implementa el Sistema de Visualización VisualProg, basado en una arquitectura desarrollada para este fin.

Actualmente existen varios sistemas destinados a facilitar el entendimiento de software, (Haibt, 1959; George, 2000; Deek y McHugh, 2002; Almeida et al., 2003; Morris, 2003; Lieberman y Fry, 2004; Deek y Espinosa, 2005; Myers, 2005; Stasko, 2007; aiSee, 2009), sin embargo, en muchas situaciones no está definido cómo las teorías cognitivas, estrategias de visualización y extracción de la información son plasmadas en esas herramientas. Además, esas aplicaciones están centradas en analizar y presentar el código fuente del programa limitando la comprensión de programas a la inspección de código, desechando el diseño de métodos flexibles que faciliten la construcción de SVP basados en componentes interrelacionados que puedan ser modificados o adaptados a diferentes contextos o propósitos. Es por ello que como parte de esta investigación se propone la arquitectura para el diseño de SVP (Arq_VP), basado en tres capas: Analizador de Código, Controladora y Vista.

2.1.3. Aplicación de Arq_VP en el diseño e implementación del SVP VisualProg

El SVP VisualProg está basado en la arquitectura Arq_VP. En la figura 2 se muestran los módulos y componentes de cada una de las capas.

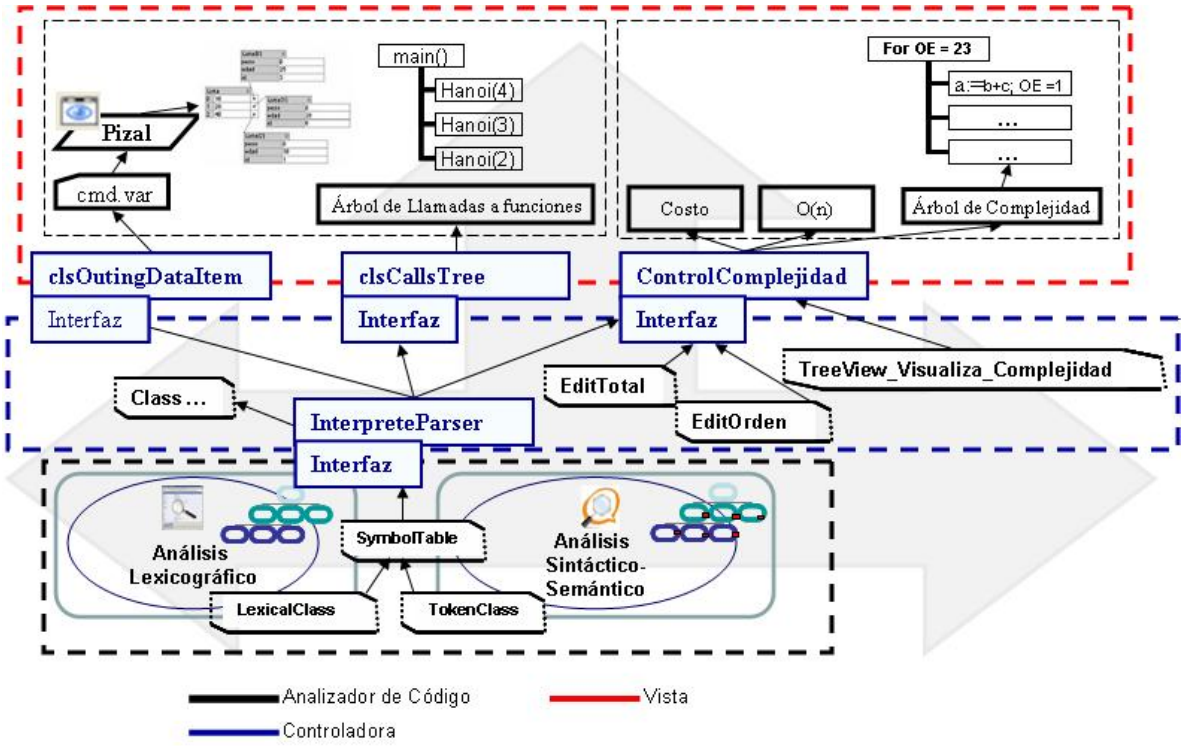

Figura 2. Arquitectura de VisualProg basada en Arq_VP

2.1.3.1 Capa Analizador de Código de VisualProg

Se encuentra el intérprete de SubC, un nuevo lenguaje desarrollado para escribir los programas que sirven de entrada al SVP (VisualProg). Este último interpreta el código SubC y suministra la información a la capa controladora que a su vez llama a los módulos de Vista, los que realizan la salida fundamental del sistema o sea la visualización. El intérprete de SubC está basado en una gramática que acepta un lenguaje sencillo.

2.1.3.2 Capas Controladora y Vista de VisualProg

En la capa controlador se tienen en cuenta diferentes acciones para apoyar la visualización de programas en SubC, está compuesta por los módulos clsOutingDataItems y ControlComplejidad.

Controlador para la visualización de datos

En el módulo clsOutingDataItems las funciones se centran en efectuar chequeos semánticos y almacenar información que será utilizada en el proceso de interpretación – visualización. De modo general se puede decir que la responsabilidad de la clase clsOutingDataItem es obtener los atributos del elemento que se desea mostrar para luego formar la cadena que será la salida del compilador y a la vez la entrada del Pizal.

En esencia Pizal es un intérprete de comandos de visualización capaz de representar, gráficamente, datos que pueden estar o no asociados a programas. La aplicación está dividida en dos partes fundamentales consistentes en una barra de herramientas en la parte superior y un área de visualización que ocupa el resto de la ventana (Figura 3).

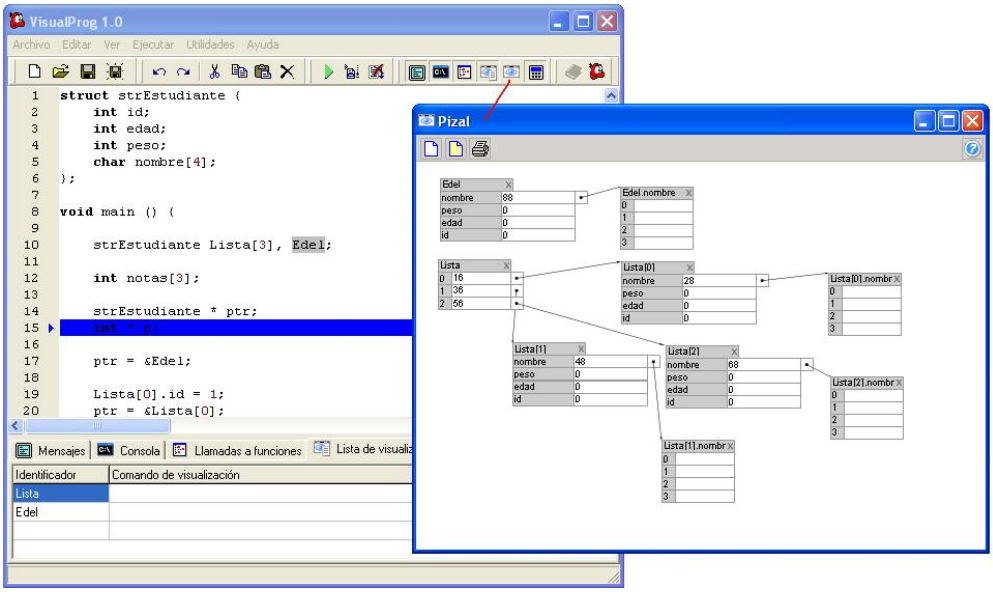

Figura 3. Vista de datos a través del sistema Pizal

Controlador para el cálculo y visualización de la complejidad del programa.

El reporte final del cálculo de la complejidad luego de la ejecución del código es el resultado de los procesos realizados en la capa controladora se muestra el siendo el tiempo de ejecución (T(n)) y el orden de complejidad del algoritmo escrito por el usuario de orden lineal O(n), así como el árbol de complejidad que se corresponde con la información detallada del valor de cada OE. Unido a este reporte, si el usuario selecciona una línea del formulario, se le muestra la información relacionada con la regla por la cual se ha llegado a este valor (Figura 4).

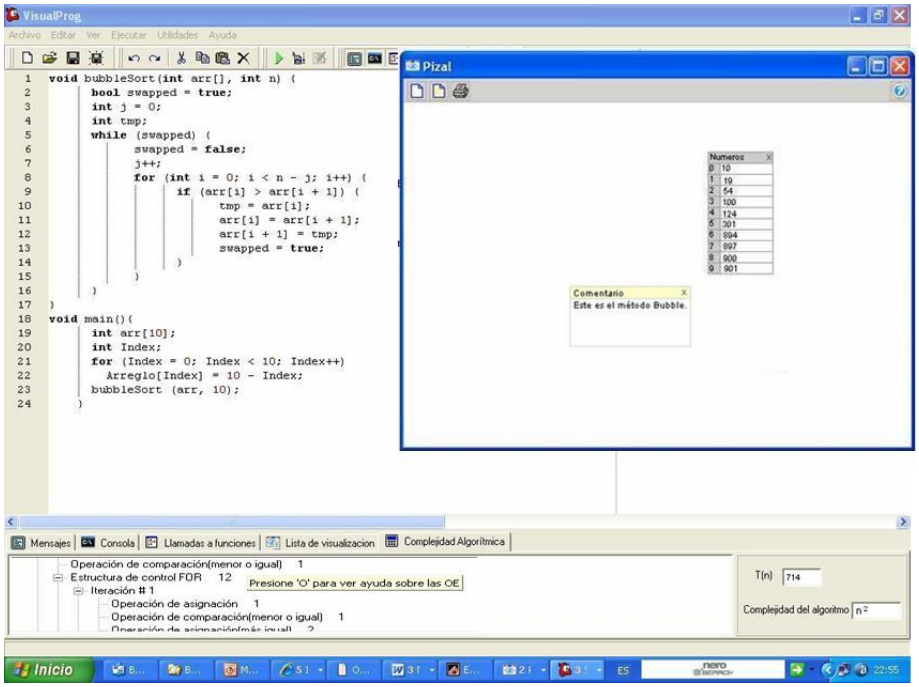

Figura 4. Reporte del cálculo de complejidad de un programa

3.1 VIA-ED en el proceso de enseñanza-aprendizaje de la asignatura Estructura de Datos

La aplicación de VIA-ED en el proceso de enseñanza-aprendizaje de la asignatura, facilita la actualización de los contenidos científicos presentados y estimula la atención y participación de los estudiantes. La atracción visual que ejerce y su contribución a la comprensión de procesos de un alto nivel de abstracción es una fuente de motivación. Sin embargo, la simple incorporación de estas tecnologías innovadoras no garantiza la efectividad de los resultados, que se logran, no tanto por el cambio de materiales, sino por la oportunidad que reportan para el cambio en la filosofía de enseñanza-aprendizaje. En esta investigación se considera que los cambios en los medios de enseñanza deben ir acompañados de profundos cambios metodológicos. El modelo propuesto como solución en esta investigación apoya el estudio independiente y participativo, fundamentalmente, para la comprensión y visualización de procesos abstractos. En su estructura se tiene en cuenta el momento de las clases y las actividades de autopreparación de los estudiantes en ambas modalidades de estudio (Figura 5).

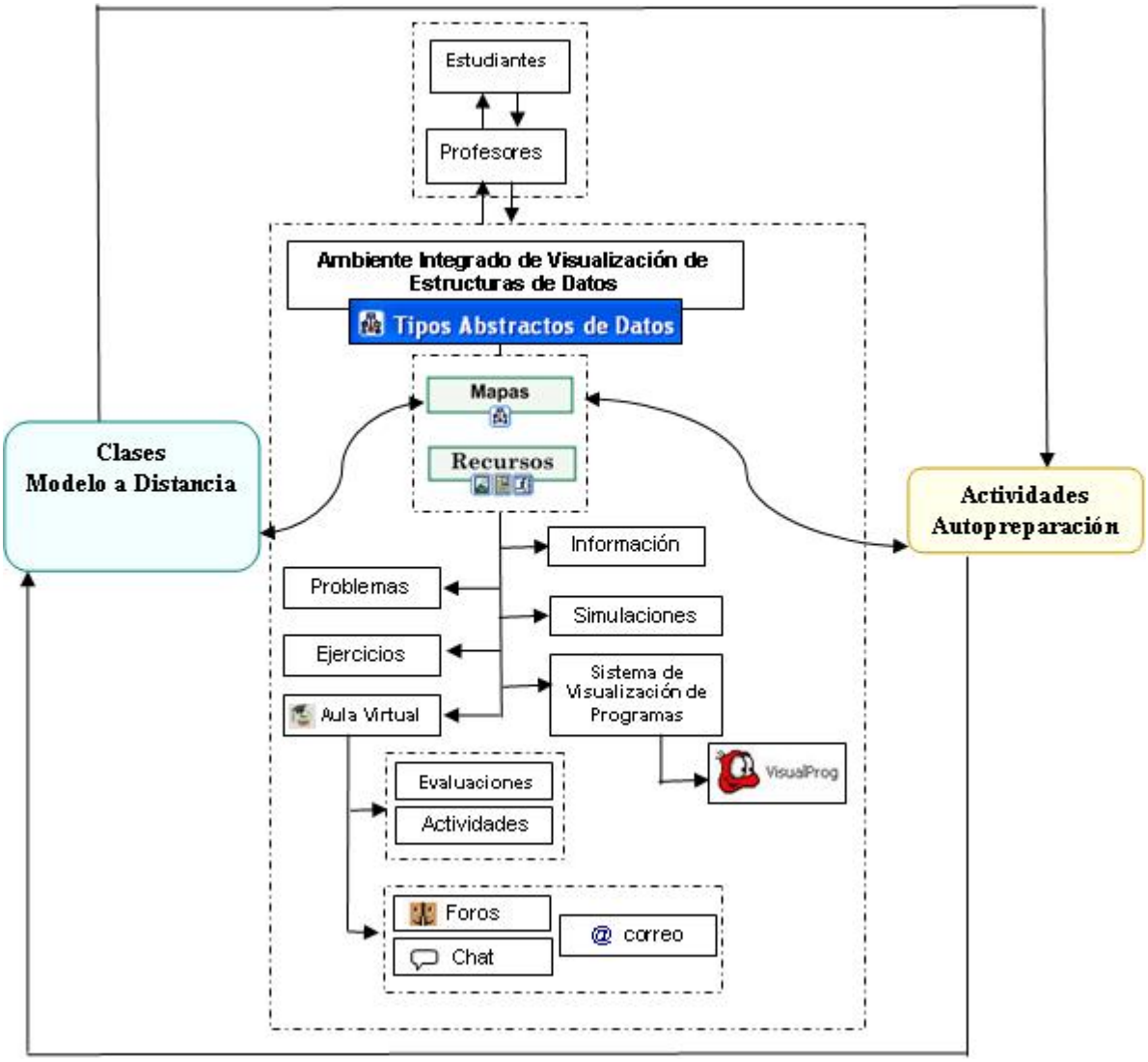

Figura 5. Modelo de interacción en VIA-ED

3.2 Validación de los resultados.

Se consultaron a dos grupos de estudiantes, el primero formado por 41 estudiantes de segundo año de la carrera Ingeniería Informática de la Universidad de Granma que utilizaron VIA-ED en el proceso de enseñanza-aprendizaje de la asignatura Estructuras de Datos durante el primer semestre del curso 2008-2009. El segundo grupo lo formaron 47 estudiantes que ya habían recibido la asignatura en años anteriores y a los que se les mostró VIA-ED, sus recursos y la forma de usarlo en el proceso de enseñanza-aprendizaje. En el caso de las variables analizadas extensibilidad, representación e impacto de VIA-ED, no se encuentran diferencias significativas entre los grupos, lo mismo sucede al analizar los resultados de la comparación entre las variables que miden la repercusión del ambiente, la importancia del SVP VisualProg y la facilidad de mostrar la traza de ejecución de un programa. Estos resultados permiten comprobar que:

- la integración de diferentes técnicas y herramientas de visualización en un ambiente de apoyo al proceso de enseñanza-aprendizaje de la asignatura Estructuras de Datos puede ser generalizado a otras áreas del conocimiento,
- su uso no requiere un entrenamiento previo, es fácil de usar, sobre todo si se tiene en cuenta que la interfaz en forma de mapa conceptual organiza los contenidos y permite la navegación de los conceptos más generales a los más inclusivos,
- los estudiantes reconocen el SVP como una herramienta que contribuye a la comprensión de conceptos y procesos de gran nivel de abstracción.

Análisis de la encuesta a los expertos

Se utilizó el método Delphi (Konow y Pérez, 2004) para la selección de los expertos, en el Anexo 9 aparecen los aspectos que se midieron para evaluar su nivel de competencia. Para elegir la muestra se tuvieron en cuenta a expertos de más de 10 años de experiencia como profesionales y profesores de la disciplina Programación de Computadoras, con dominio en el uso de Técnicas de Visualización, en metodología de la enseñanza de la programación y en el diseño de software educativo.

Los expertos plantearon satisfacción con la propuesta, destacándose criterios muy positivos en relación con la repercusión del ambiente, la integración de los recursos y el nivel de significación que se le otorga al uso de mapas conceptuales como organizadores del conocimiento, el SVP VisualProg, los textos explicativos y al Aula Virtual. En el caso de VisualProg, se resalta la influencia que tuvo seguir la traza de ejecución del programa y comprobar el efecto que provoca en los datos, además, de permitir la implementación de cualquier algoritmo. Por lo que se considera muy adecuado en impacto de VIA-ED en el desarrollo del proceso de enseñanza-aprendizaje de la asignatura Estructuras de Datos de la carrera de Ingeniería Informática en la Universidad de Granma.

4. CONCLUSIONES Y TRABAJO FUTURO

Los resultados obtenidos permiten concluir que:

- La integración de diferentes Técnicas de Visualización de Programas favorece el desarrollo de ambientes que apoyan el aprendizaje y contribuyen a la solución de los problemas tradicionales de los programadores noveles relacionados con el diseño, implementación y comprensión de las estructuras de datos y sus operaciones.
- El Ambiente Integrado de Visualización de Estructuras de Datos basado en mapas conceptuales, facilita la visualización de los conceptos relacionados con las estructuras de datos, sirviendo como un repositorio de aplicaciones, de recursos de información y comunicación que contribuye a la comprensión y diseño de programas.
- La Arquitectura (Arq_VP) apoya el diseño de SVP, delimita los procesos y métodos a utilizar en cada capa, así como las interfaces de conexión entre ellas, propone el empleo en la capa Controladora de métodos de extracción de información tanto estáticos (Técnicas de compilación) como dinámicos (a través del Esquema de Instrumentación de Código y la Técnica de control de las iteraciones para optimizar la cantidad de información extraída). Facilita la inclusión o modificación de interfaces y procesos al SVP sin que se afecte el sistema.

- El SVP dinámico VisualProg basado en Arq_VP ayuda al diseño de estructuras de datos y al análisis de la complejidad de programas escritos en el lenguaje SubC. El desarrollo de este lenguaje facilitó la recuperación, manejo y transformación de la información del programa que se utilizó en las Capas Analizador de Código, Controladora y Vista del sistema. En general, el SVP VisualProg, es un recurso importante de VIA-ED, porque permite al usuario implementar sus algoritmos y facilita la toma de decisiones en el diseño de los datos.
- El ambiente VIA-ED se aplicó en el proceso de enseñanza-aprendizaje de la asignatura Estructura de Datos de la carrera de Ingeniería Informática en la Universidad de Granma en el curso 2008-2009 y fue evaluado satisfactoriamente por estudiantes y expertos.

Derivadas del estudio realizado, así como de las conclusiones de este trabajo, se recomienda:

- Implementar ambientes similares que aborden otros temas de la disciplina Técnicas de Programación de Computadoras, en particular en el área de Técnicas de Compilación, que pueden usar el Sistema VisualProg y el diseño del lenguaje SubC como materiales de estudio.
- Realizar un análisis del Plan de Estudios D de la carrera Ciencias de la Computación para que la herramienta profundice en los contenidos abordados en la enseñanza de esa ciencia.
- Dotar al ambiente de Técnicas de Inteligencia Artificial que permitan una navegación basada en el modelo de conocimiento de los estudiantes.

5. REFERENCIAS BIBLIOGRÁFICAS

ACM y IEEE-CS. (2005). Computing Curricula 2005. The Overview Report covering undergraduate degree programs in Computer Engineering, Computer Science, Information Systems Information Technology, Software Engineering: ACM & IEEE-CS, 58.

aiSee. (2009). aiSee Application Examples: PAG/WWW (Version 2.0) [Software Educativo].

Almeida, F., Blanco, V. y Moreno, L. (2003). EDApplets: Una Herramienta Web para la Enseñanza de Estructuras de Datos y Técnicas Algorítmicas. Artículo presentado en X Jornadas de Enseñanza Universitaria de la Informática, Universidad de Laguna. Tenerife, 12.

Anderson, J. (1983). The architecture of cognition (Harvard University Press ed. Vol. 2). Cambridge, MA, 63-78 pp.

Brown, G., Carling, R., Herot, C., Kramlich, D. y Souza, P. (2005). Program Visualization: Graphical Support for Software Development. IEEE Computer, 18(8), 27-35.

Clinton, H. (2004). Program Monitoring and Visualization (Springer-Verlag ed. Vol. 2). Denver, 213 pp.

Chang, S., Tauber, M., Yu, B. y Yu, J. (2008). A Visual Language Compiler. IEEE Transactions on Software Engineering, 506-525.

Deek, F. P. y Espinosa, I. (2005). An evolving approach to learning problem solving and program development: The distributed learning model. International Journal on E-

Learning, 4(4), 409-426.

Deek, F. P. y McHugh, J. (2002). SOLVEIT: An experimental environment for problem solving and program development. Journal of Applied Systems Studies, 2(2), 376-396.

Deek, F. P., McHugh, J. y Hiltz, S. R. (2008). Methodology and technology for learning programming. Systems and Information Technology, 4(1), 25-37.

Frías, I., Soler, Y. y Lezcano, M. (2007). Sistema de Enseñanza Asistida por Computadora para la visualización de operaciones sobre estructuras de datos y animación de algoritmos. Máster. Maestría en Nuevas Tecnologías para la Educación, Universidad de Granma, Bayamo, 74.

George, C. (2000). EROSI - Visualizing recursion and discovering new errors. Artículo presentado en ACM SIGCSE Technical Symposium on Computer Science Education, 305-309.

Graf, M. (2007). A Visual Environment for the Design of Distributed Systems. Workshop on Visual Languages. IEEE Computer Society, pp. 330-344.

Grinstein, L. y Levkowits, K. (2005). Perceptual Issues in Visualization. New York: Springer-Verlag, 59 pp.

Gugerty, L. y Olson, G. (2006). Debugging by skilled and novice programmers. Artículo presentado en ACM SIGCHI on Human Factors in Computing Systems, 171-174.

Haibt, L. (1959). A Program to Draw Multi-Level Flow Charts. Artículo presentado en Western Joint Computer Conference, San Francisco, CA, 131-137.

Hundhausen, C. (2006). Integrating algorithm visualization technology into an undergraduate algorithms course: ethnographic studies of a social constructivist approach (MIT-Press ed.). Blankenburg, 27 pp.

Jeffery, C. (2006). Program Monitoring and Visualization (Springer-Verlag ed.). Baja California, 101 pp.

Kearns, M. y Vazirani, U. (2004). An Introduction to Computational Learning Theory: The Bactra Review. Occasional and eclectic book reviews by Cosma Shalizi. How to Build a Better Guesser, 32 pp.

Konow, I. y Pérez, G. (2004). Método DELPHI, Métodos y Técnicas de Investigación Prospectiva para la toma de Decisiones (Vol. I, pp. 26). Fundación de Estudios Prospectivos (FUNTURO). Chile

Lieberman, H. y Fry, C. (2004). Bridging the gulf between code and behavior in programming. Artículo presentado en In ACM Conference on Computers and Human Interface, Denver, Colorado, 9.

Maimone, M. W., Tygar, J. y Wing, J. (2008). Miro Semantics for Secu rity. IEEE Workshop on Visual Languages, pp. 45-51.

Moroni, N. y Señas, P. (2006). Hypermedial Conceptual Mapping: A Development Methodology. Artículo presentado en 13th International Conference on Technology and Education, University of Texas at Arlington, Departament of Computer Science an Engineering. New Orleans, 5.

Morris, J. (2003). Algorithm Animation: Using algorithm code to drive an animation. Journal

of Visual Languages and Computing, 6.

Moshell, M., Hughes, C., Lacy, L. y Lewis, R. (1987). A Spreadsheet-Based Visual Language for Freehand Sketching of Complex Motions. Workshop on Visual Languages. IEEE Computer Society, pp. 94-104.

Myers, B. (2004). Taxonomies of Visual Programming and Program Visualization (School of Computer Science. Carnegie Mellon University ed.). Pittsburgh, PA, 33 pp.

Myers, B. (2005). Incense: A System for Displaying Data Structures. Artículo presentado en Computer Graphics: SIGGRAPH '05, 115-125.

Naps, T. y Rößling, G. (2004). Evaluating the Educational Impact of Visualization (pp. 124-136): Addison-Wesley Company.

Newell, A. (1972). Human problem solving (Vol. 1). Englewood Cliffs, NJ: Prentice Hall, 49 pp.

Price, B. A., Baecker, R. M. y Small, I. S. (2004). A Principled Taxonomy of Software Visualization. Journal of Visual Languages and Computing, 4 (3), 211-266.

Satratzemi, M., Dagdilelis, V. y Evageledis, G. (2004). A system for program visualization and problem-solving path assessment of novice programmers. Artículo presentado en Annual Joint Conference Integrating Technology into Computer Science Education, 6th Annual Conference on Innovation and Technology in Computer Science Education, 137-140.

Señas, P. y Moroni, N. (2006). Herramientas no convencionales para la enseñanza de la programación. Doctor en Ciencias. Doctorado en Computación, Instituto de Ciencias e Ingeniería de Computación Universidad Nacional del Sur, Bahía Blanca-Argentina, 112.

Soler, Y. y Lezcano, M. (2007). Organización del conocimiento de la asignatura Programación II para Ingeniería Informática basada en mapas conceptuales. Tesis de Maestría, 75.

Stasko, J. (2007). PolkaW Animation Designer's Package (Version 4.0) [Programa de Computadora].

Stasko, J., Domingue, J., Brown, M. y Price, B. (2006). Software Visualization: Programming as a Multimedia Experience. MIT Press, 28.

TechnM. (2005). Visual Mind - eficciency througth smart thinking. Disponible en: http://www.visual-mind.com/

Yolanda Soler Pellicer, nace en 1967, se gradúa de Licenciada en Cibernética Matemática en 1990, cursa la Maestría de Computación Aplicada y defiende su tesis en 2007, se encuentra desarrollando su tesis para optar por el título de Doctor en Computación y Automática, estos estudios se han desarrollado en la Universidad Central de Las Villas (UCLV), Cuba. Es profesora Auxiliar de la Universidad de Granma, Cuba, ha participado en más de 20 eventos nacionales e internacionales, Investiga en la línea de Visualización de Algoritmos y el Cálculo de la Complejidad. Forma parte del Comité de Referato de la Revista COGNICION, ISSN: 1850-1974. Es miembro de la Sociedad Cubana de Matemática y Computación.

Mateo Lezcano Brito, nace en el año 1949, se gradúa de Licenciado en Cibernética Matemática en el año 1982, hace una Maestría de Computación Aplicada en 1995 y obtiene su título de Doctor en Ciencias Técnicas en 1998. Todos sus estudios los realiza en La Universidad Central de Las Villas (UCLV), Cuba. Actualmente es Profesor Titular del departamento de Ciencia de la Computación de la UCLV y fue director del Centro de Estudios de Informática.

Isvani Frías Blanco, nace en 1981, se gradúa de Licenciado en Ciencia de la Computación en el año 2005 en la Universidad de Oriente, Cuba. Realiza la maestría en Nuevas Tecnologías para la Educación en la Universidad de Granma y defiende su tesis en 2007, comienza sus estudios doctorales en 2008 en la Universidad Central de las Villas, vinculado a los temas relacionados con las técnicas de Soft Computing para la Visualización de Software y Técnicas de Inteligencia Artificial.

Edel Ángel Rodríguez Sánchez, nace en el año 1980, se gradúa en el año 2007 como Ingeniero Informático en la Universidad de Granma, Cuba, fue Alumno de Alto Aprovechamiento Docente, forma parte del grupo de investigación sobre Visualización de Programas y Algoritmos y análisis de complejidad.

Manuel Alarcón Suárez, nace en 1985, se gradúa en el año 2008 como Ingeniero Informático en la Universidad de Granma, Cuba, fue Alumno de Alto Aprovechamiento Docente, forma parte del grupo de investigación sobre Visualización de Programas y Algoritmos y análisis de complejidad.

Manuel José Linares Álvaro, nace en 1969, se gradúa de Ingeniero en Ciencias Agrícolas en 1991 en la Universidad de Granma, Cuba, cursa la Maestría en Computación Aplicada de la Universidad “Marta Abreu” de Las Villas y defiende su tesis en el año 2007. Dirige la red de computadoras de la Universidad de Granma, es profesor de la disciplina de Redes y

Sistemas Operativos, temas en los cuales investiga, fundamentalmente en los relacionados con la instalación del protocolo IPv6.