AMBIENTE INTEGRADO DE VISUALIZACIÓN DE ESTRUCTURAS DE DATOS PARA LA ENSEÑANZA-APRENDIZAJE DE LA PROGRAMACIÓN DE COMPUTADORAS

INTEGRATED ENVIRONMENT OF VISUALIZATION OF DATA STRUCTURES FOR THE TEACHING-LEARNING OF COMPUTER PROGRAMMING

Yolanda Soler Pellicer[1] (yoly@udg.co.cu)

Karina Virginia Mero Suárez[2] (karinaunesum@yahoo.com)

Edwin Joao Merchán Carreño[3] (joaounesum@yahoo.es)

RESUMEN

En el proceso de enseñanza–aprendizaje de las asignaturas de programación de computadoras, se han enfrentado a problemas en el diseño y selección adecuada de las estructuras de datos para representar la información y las operaciones básicas que con ellas se realizan, lo que dificulta la obtención de algoritmos eficientes. Con base en esta situación se desarrolla un Ambiente Integrado de Visualización de Estructuras de Datos basado en mapas conceptuales que contiene un repositorio de recursos, en el que se destaca el sistema VisualProg que tiene como entrada el código en el lenguaje SubC y cuenta con los componentes de visualización de código, de datos, del árbol de recursividad y del análisis de complejidad del programa. La implementación se fundamentó en una arquitectura concebida en tres capas: Analizador de Código, Controladora y Vista. Se utilizó la modelación para el desarrollo de los algoritmos, métodos estadísticos para comprobar la utilidad de la propuesta y validar los aportes fundamentales de la investigación. Se aplicó en el proceso de enseñanza-aprendizaje de la asignatura Estructura de Datos de la carrera Ingeniería Informática en la Universidad de Granma, Cuba, e Ingeniería en Sistemas de la Universidad Estatal del Sur de Manabí en Ecuador. Se demuestró que la integración de técnicas de visualización contribuye a solucionar problemas relacionados con el diseño e implementación de estructuras de datos y programas.

PALABRAS CLAVE: Sistemas de visualización de programas, estructuras de datos, mapas conceptuales.

ABSTRACT

In the teaching-learning process of computer programming subjects, problems have been faced in the design and adequate selection of data structures to represent the information and the basic operations that are carried out with them, which makes it difficult to obtain efficient algorithms. Based on this situation, an Integrated Environment for Visualization of Data Structures is developed based on conceptual maps that contains a resource repository, in which the VisualProg system is highlighted, which has the code in the SubC language as input and has the components of code visualization, of data, of the recursion tree and of the complexity analysis of the program. The implementation was based on an architecture

[1] Doctora en Ciencias Técnicas. Profesora Titular. Departamento de Informática. Universidad de Granma, Bayamo, Cuba.

[2] Profesora de la carrera Ingeniería en Sistemas Computacionales. Universidad Estatal del Sur de Manabí, Ecuador.

[3] Profesor de la carrera Ingeniería en Sistemas Computacionales. Universidad Estatal del Sur de Manabí, Ecuador.

conceived in three layers: Code Analyzer, Controller and View. Modeling is used to develop the algorithms, statistical methods to verify the usefulness of the proposal and validate the fundamental contributions of the research. The environment was applied in the teaching-learning process of the subject Data Structure of the Computer Engineering career at the University of Granma, Cuba and Systems Engineering of the Southern State University of Manabí in Ecuador. It is demonstrated that the integration of visualization techniques contributes to solve problems related to the design and implementation of data structures and programs.

KEY WORDS: Program visualization systems, data structures, concept maps.

La tarea de aprender a usar el conjunto de símbolos asociado a un Lenguaje de Programación (LP) conforme a una sintaxis, relacionándolos con una semántica, demanda un esfuerzo considerable para los alumnos de los primeros años de las carreras de perfil informático. A esto se suma, en muchos casos, una formación previa deficiente que les dificulta organizar nuevos conceptos para construir taxonomías y diferenciar propiedades que permitan establecer pautas para razonar sobre ellas. Esta situación puede agravarse en un modelo de enseñanza a distancia si no se traza una estrategia adecuada.

En función de ello, se han usado diferentes vías y herramientas que ayudan a mejorar el aprendizaje y desarrollo de algoritmos, sin embargo 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 es la de visualización que, en general, consiste en el uso de recursos gráficos, de animación y multimedios, con una importante interacción entre el usuario y la computadora (Chang y otros, 2012; George, 2010; Hundhausen, 2006; Hyrskykari y Raiha, 2007; Kelly y Keller, 2005; Moor y Deek, 2012; Moriconi y Hare, 2015; Moroni y Señas, 2015b; Stasko, 2011).

Independientemente de las facilidades que los lenguajes de programación modernos ofrecen a los programadores, no todos muestran el efecto que el código escrito provoca en los datos, la visualización del árbol de llamadas a una función recursiva o de presentar el cálculo de la complejidad del programa. En los últimos tiempos, se ha trabajado en el desarrollo de entornos específicos para ayudar y conducir a los programadores noveles en el complejo proceso de aprender a programar. De ahí la importancia de la visualización de software, que comprende la de algoritmos (permite mostrar abstracciones de alto nivel que lo describen) y la de programas (visualiza el código real del programa y las estructuras de datos que utiliza). Ambas pueden mostrarse en forma estática o dinámica (Clinton, 2014; Stasko, 2012; Stasko y Lawrence, 2014).

En consecuencia, las técnicas de visualización de algoritmos y programas se han explotado y desarrollado ampliamente, lo que ha generado el surgimiento de múltiples sistemas que se ajustan a diversas categorías porque muestran diferentes modos de visualización y manejo. Entre estos se encuentran TANGO de Stasko (2010), MacGnome de Chandhok (2010), Incense implementado por Myers (2010), Diagramas Orientados a Objetos por Cunningham y Beck (2011), TPM elaborado por Eisenstadt y Brayshaw (2011), Sorting and Sorting desarrollado por Baecker (2011), VisualizationTool diseñado por Deek (2013) y BALSA de Brown y Sedgewick (2014).

En este contexto, Stojanovic (2002) y Chestlevar (2015), proponen el uso de mapas conceptuales para la enseñanza de conceptos básicos de programación y desarrollo de

algoritmos. Respecto a las destrezas cognitivas, los mapas conceptuales favorecen las conexiones con ideas previas, la capacidad de inclusión, la diferenciación progresiva entre conceptos, la integración, asimilación y presentación de nuevas relaciones entre ellos (Estrada y Febles, 2016; Liens, 2004).

El propósito del ambiente de enseñanza-aprendizaje que se presenta en este trabajo es el de ayudar a los estudiantes a mejorar el diseño de las estructuras de datos y programas, facilitar el análisis de eficiencia al mostrar, de forma dinámica, el cálculo de la complejidad, del comportamiento del código y los datos, así como apoyar la comprensión de procesos de un alto nivel de abstracción. Por otra parte, permitirá el trabajo colaborativo y la interacción entre los sujetos del proceso de enseñanza-aprendizaje, a través de la integración de diversas técnicas de visualización y de un conjunto de recursos entre los que se encuentra un Aula Virtual, simuladores, mapas conceptuales y el Sistema de Visualización de Programas VisualProg, apoya, además, el proceso de enseñanza y asesoría a los programadores noveles.

La visualización de software presenta un medio audio-visual, que facilita la interactividad cuyas bondades en el campo de la psicopedagogía han sido experimentadas. Algunos autores como Bladek y Deek (2013), Guttag y Liskov (1986), Hennessy (2011), Morris (2012), Stasko (2011), Stasko y Lawrence (2014), consideran que los sistemas no solo deben estar diseñados por especialistas en computación sino por psicopedagogos que permitan establecer un equilibrio entre el uso de potentes recursos informáticos y lo que el ser humano puede percibir efectivamente de ellos. La visualización de algoritmos consiste en mostrar abstracciones de alto nivel que describen el algoritmo, y la de programas se refiere al código real de programa y a las estructuras de datos. Ambas pueden darse en forma estática o dinámica (Frías, Soler y Lezcano, 2014).

Las herramientas que realizan análisis estático examinan el texto y proveen información sobre el programa que es válida para todas las ejecuciones, independientemente de los valores de los datos de entrada. Las técnicas de análisis dinámico emplean editores de sintaxis, optimizadores de código, embellecimiento de la exhibición del código, entre otras facilidades (Clinton, 2014).

Múltiples sistemas citados por Chang y otros (2012), Graf (2009), Moshell y otros (2007), usan los gráficos para mostrar algunos aspectos del programa. Mientras que Myers (2011), Price, Baecker y Small (2014), Satratzemi, Dagdilelis y Evageledis (2010), presentan otros Sistemas de Visualización de Programas (SVP) 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.

Para Naps y Rößling (2014), la visualización de la conducta dinámica de los programas, así como sus modelos abstractos, los algoritmos, tienen un impacto psicopedagógico en la enseñanza. Se han experimentado las facilidades que en el campo de la psicopedagogía tiene la visualización de software como un medio audio-visual interactivo multimedial. Diversos autores entre los que se encuentran Bladek y Deek (2013) y Silva y Hernández (2013) han identificado y clasificado gran cantidad de problemas que los principiantes afrontan al aprender a programar a través de investigaciones realizadas en el área de la psicología, la interacción hombre-máquina, la cognición y la pedagogía.

Por otra parte, Liffick y Aiken (2011), como pedagogos, plantean que los programadores noveles pueden tener dificultades al entender un nuevo concepto, no porque sea difícil, sino porque depende de un concepto anterior y esto no se ha tenido en cuenta por el docente. Otro aspecto puede ser atribuido a la inadecuada selección del problema a solucionar para alcanzar determinada habilidad. Lim (2012), plantea, por consiguiente, que los nuevos programadores comienzan a escribir programas y generar el código sin realizar un proceso de análisis planificado y organizado. Por su parte, Moor y Deek (2012), así como, Pane y Myers (2013), Bennedsen y Caspersen (2013), observan que las estrategias de solución al problema general deberían ser explícitamente adquiridas a través de habilidades en el desarrollo de programas.

Desde la óptica psicológica, Guzdial y Elliott (2012), valoran que la manera en que la información está representada y se manipula en una computadora provoca un desafío a la comprensión. Como resultado, los estudiantes tienen un inadecuado modelo mental para comprender el funcionamiento interno de la computadora. Algunos de los conceptos de programación son abstractos por naturaleza, y otros no tienen una analogía en el mundo real. También hay que tener en cuenta que, en la mayoría de los casos, los nuevos programadores han sido excelentes usuarios de aplicaciones y no se han preparado psicológicamente para enfrentar un proceso de creación de aplicaciones y sus dificultades intrínsecas.

Estas consideraciones pedagógicas y psicológicas, a las que se integran las bases tecnológicas, planteadas por Deek (2013), corroboran que las herramientas que usan la visualización para ayudar a los estudiantes en el proceso de desarrollo de un programa facilitan el aprendizaje y la modelación de conceptos y procesos, el análisis de programas y la solución de problemas, de ahí la necesidad de desarrollar una propuesta que apoye el proceso de enseñanza-aprendizaje de la asignatura Estructura de Datos.

Metodología y herramientas usadas

Para desarrollar el Ambiente Integrado de Visualización de Estructuras de Datos (VIA-ED), los recursos que lo componen y validar los resultados de su aplicación en la práctica docente de la carrera Ingeniería Informática de la Universidad de Granma, Cuba e Ingeniería en Sistemas de la Universidad Estatal del Sur de Manabí en Ecuador, se utilizaron diferentes herramientas. Entre ellas el cmapTools, que permite desarrollar el ambiente que integra en un mapa los conceptos de la asignatura Estructura de Datos (Cañas y otros, 2009; Cañas y Novak, 2012). Se diseñó además, una arquitectura para el desarrollo de Sistemas de Visualización de Programas (Arq_VP) que es utilizada en la implementación del Sistema de Visualización dinámica de Programas escritos en el lenguaje SubC, también desarrollado como parte de la investigación.

Para evaluar el Ambiente Integrado de Visualización de Estructuras de Datos (VIA-ED) se realizaron encuestas a estudiantes de ambas carreras y se consultó el criterio de 45 expertos cubanos y ecuatorianos en Programación de Computadoras que tuvieran competencias sobre el uso de Técnicas de Visualización de Programas, profesores de Programación con conocimiento sobre metodología de enseñanza de la Programación y el diseño de Software Educativo.

Se consultaron dos grupos de estudiantes, el primero formado por 41 estudiantes de segundo año de la carrera Ingeniería Informática que utilizaron VIA-ED en el proceso de

enseñanza-aprendizaje de la asignatura Estructuras de Datos durante el primer semestre del curso 2014-2015. El segundo grupo lo formaron 88 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.

Las encuestas miden siete variables: repercusión, recursos, traza, visión, extensibilidad, representación, impacto, que permiten validar el comportamiento del ambiente. Los resultados fueron sometidos a diferentes pruebas con el paquete estadístico SPSS (IBM, 2014).

El protagonismo de las técnicas de información y las comunicaciones en los ambientes de aprendizajes, ha hecho que el concepto tradicional de software educativo haya migrado, de forma continua, a nuevas y variadas formas, muchas de ellas basadas en el uso de redes de computadoras, la posibilidad de que los actores del proceso educativo se relacionen y compartan experiencias a través de plataformas web, es cada vez más común, incluso, aumenta constantemente el número de instituciones pertenecientes a países del tercer mundo, que aplican estas tecnologías (Díaz y Leal, 2013).

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

Apoyados en las técnicas de visualización y en la búsqueda de soluciones a las deficiencias encontradas en el diseño y selección adecuada de las estructuras de datos para representar la información y las operaciones básicas que con ellas se realizan, lo que dificulta la obtención de algoritmos eficientes, se propone un ambiente basado en mapas conceptuales, que organiza e integra el modelo del conocimiento de esta materia y sirve de interfaz visual y repositorio de recursos informáticos (un Aula Virtual, simulaciones, animaciones, imágenes y documentos en diferentes formatos), útiles para el autoaprendizaje en un modelo de estudios semipresencial. Incluye, además, el Sistema de Visualización dinámico de Programas VisualProg, como su principal recurso de aprendizaje (Figura 1).

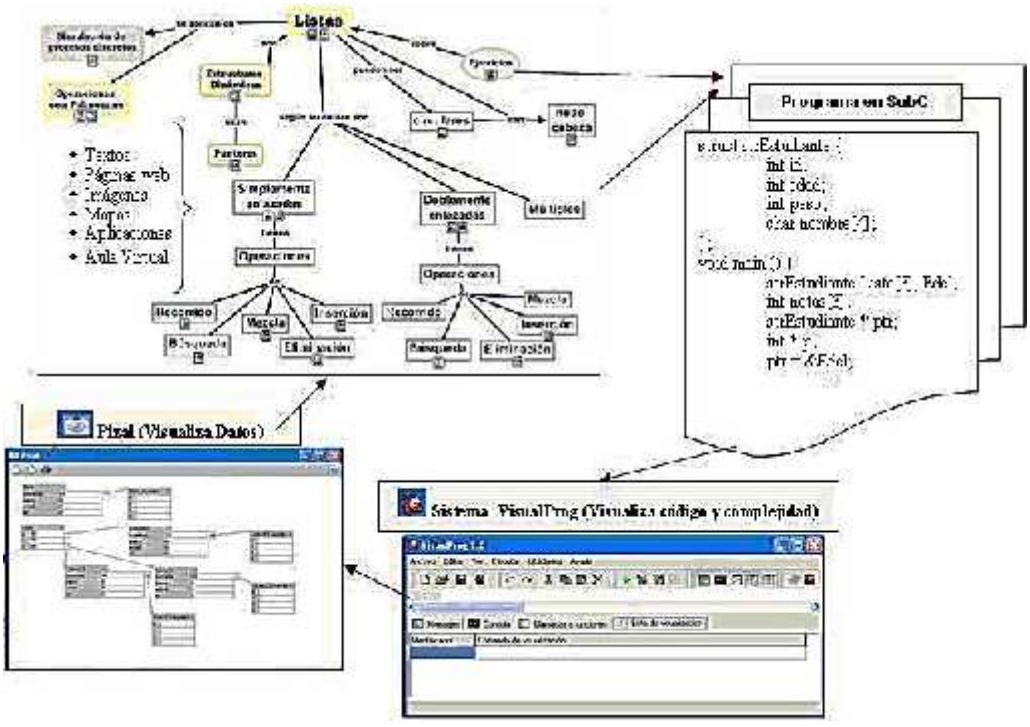

Figura 1. Esquema del Ambiente Integrado de Visualización de Estructuras de Datos

Con la herramienta cmapTools, se diseñó el mapa conceptual Tipos Abstractos de Datos, integrado por otros 24 mapas conceptuales y 204 recursos, visibles también desde Internet. El ambiente facilita la colaboración en la edición y construcción de mapas conceptuales, a través de hilos de discusión sincrónicos y asincrónicos.

- Informaciones textuales, imágenes, aplicaciones y simulaciones

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 uno se realiza un análisis de la complejidad; si son operaciones que puedan implementarse por varios métodos, como el caso del ordenamiento en arreglos, 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.

A los nodos del mapa conceptual que representan las operaciones con las estructuras de datos se le adicionan 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 representan, 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. También se insertan simulaciones asociadas a diversos procesos que almacenan la información representada en listas, arreglos y pilas, que le permiten al estudiante comprender la importancia y campos de aplicación de las estructuras de datos en la solución de problemas complejos (Cormen y otros, 2011; Hayet, 2012).

- Sistema de Visualización dinámica de Programas VisualProg

El recurso más importante de VIA-ED debe facilitar la visualización dinámica de los datos y el código de programas, mostrar las operaciones, permitir la realización de cambios en el código del programa y comprobar el efecto que provoca en los datos; por lo que se implementa el sistema VisualProg el cual, para llevar a cabo el proceso de visualización realiza un mapeo de las variables activas en un instante determinado y luego las representan a objetos de animación que sufrirán cambios en correspondencia con la creación, transformación de los valores y destrucción de las variables. VisualProg está basado en la Arquitectura para desarrollar Sistemas de Visualización de Programas (Arq_VP), realizada como parte de la investigación, la que consta de tres capas: Analizador de Código, Controladores y Vistas de código o datos.

En la Figura 2 se pueden apreciar los principales módulos del sistema. En correspondencia con la arquitectura propuesta, los objetos para la interpretación de los programas se encuentran en la capa Analizador de Código. En la capa Controladora se definen clases relacionadas con las representaciones gráficas de los objetos que se visualizarán en la capa Vista, en la cual se ubica el sistema Pizal que visualiza la transformación de los datos.

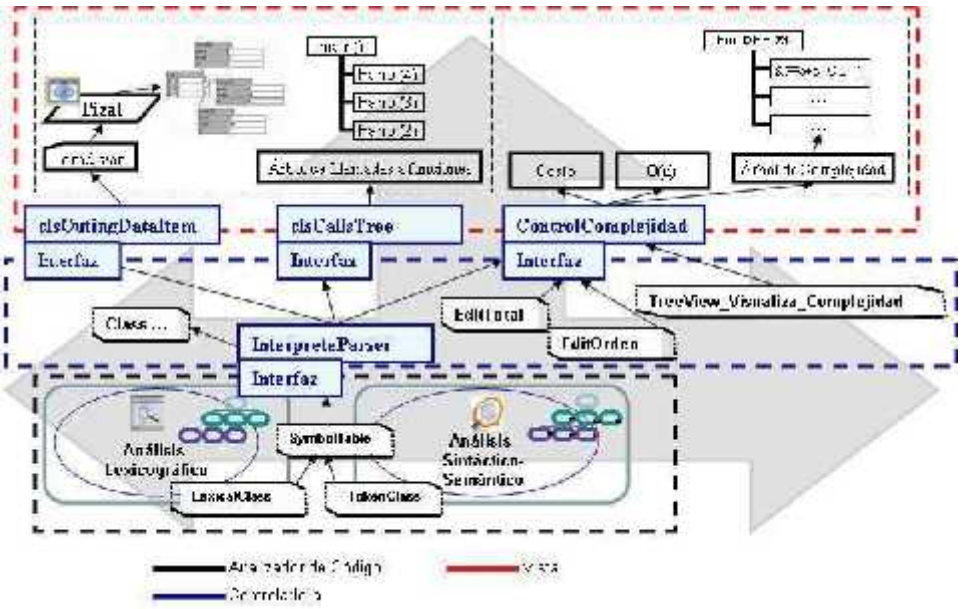

Figura 2. Arquitectura de VisualProg basada en Arq_VP

En la capa Controladora se agrega un módulo para la representación gráfica de aspectos relacionados con el cálculo de complejidad de programas y su correspondiente módulo en la capa Vista. Se incluye el procesamiento del árbol de llamadas a funciones, que apoya la comprensión de funciones recursivas, la vista en consola de los resultados de la ejecución de un programa y la de mensajes.

- Visualización de datos

El sistema Pizal, por medio de la escritura de código en ActionScript controla los elementos que se muestran de forma que se correspondan con los datos manejados por el programa. Pizal representa cuatro tipos básicos de datos, variables, punteros, arreglos, estructuras y la combinación de estos que permite la representación de estructuras de datos más complejas. La aplicación permite la adición de comentarios a la representación gráfica obtenida, la impresión de esta y reiniciar el área de visualización. Con esta facilidad el estudiante puede realizar cambios en el código del programa y visualizar el efecto que provoca en los datos en el momento de la ejecución (Figura 3).

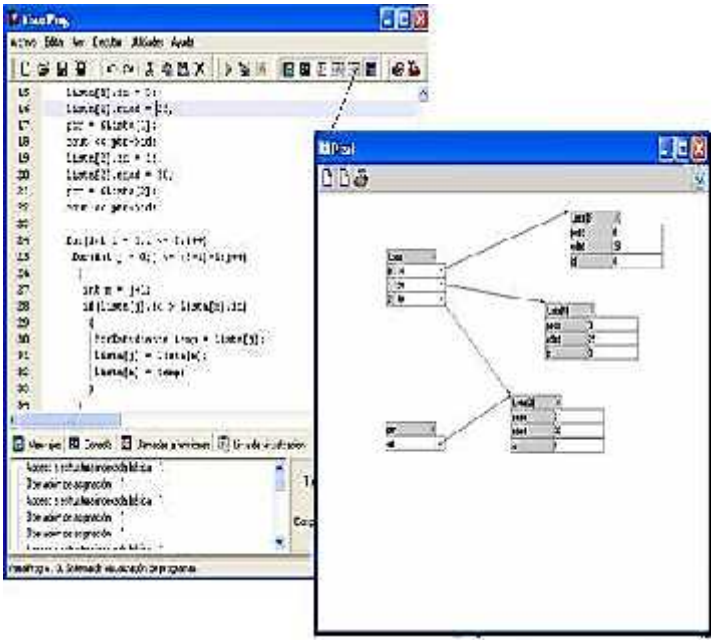

Figura 3. Vista de datos a través de Pizal

- Vista de la complejidad de los programas

La conexión del Controlador de Complejidad con el formulario principal del sistema hace posible que algunas de las funciones de la clase manipulen componentes que permiten mostrar el costo del programa y el cálculo de la función O(n), así como las reglas para el cálculo del costo de cada Operación Elemental (OE) (Figura 4).

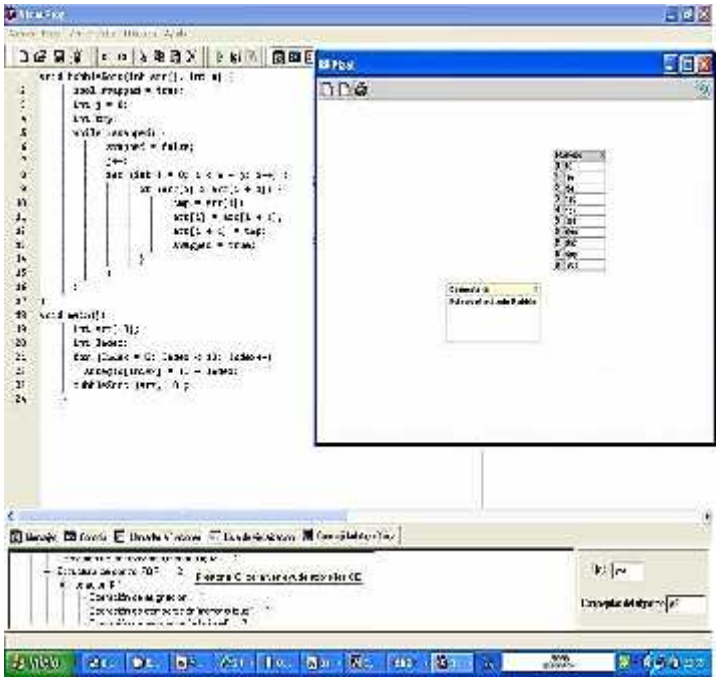

Figura 4. Vista de datos y de la complejidad a través de Pizal

- Visualización del árbol de llamadas a funciones

Uno de los problemas detectados en la comprensión de los algoritmos está relacionado con el uso de la recursividad. Para explicar este proceso sin usar VisualProg, se representaba el árbol de llamadas a las funciones de forma gráfica en la pizarra o mediante su representación estática a través de una imagen digital. El sistema VisualProg brinda la opción de visualizar el árbol de llamadas y comprobar cómo varía si se modifican los parámetros, lo que facilita no solo la comprensión, sino también la explicación de estos procedimientos.

Por último, se propone un modelo y la metodología para usar el ambiente VIA-ED como apoyo a la orientación y el estudio independiente y participativo, fundamentalmente, para la comprensión y visualización de procesos abstractos. Se tiene en cuenta la tipología de clases, el modelo de enseñanza utilizado y se proponen en cada caso cómo usar el ambiente y sus recursos.

- Validación estadística de la efectividad del VIA-ED

Para validar la efectividad de VIA-ED se realizaron encuestas a estudiantes de la carrera Ingeniería Informática en la Universidad de Granma e Ingeniería en Sistemas de la Estatal del Sur de Manabí en Ecuador, que lo usaron en el proceso de enseñanza-aprendizaje de la asignatura Estructura de Datos en el período 2014-2017. Se procesaron, además, los criterios de estudiantes que habían recibido la asignatura en cursos anteriores y no tuvieron experiencia de trabajo con el ambiente, solo se les demostraron sus facilidades para que pudieran expresar sus valoraciones sobre la incidencia que hubiera tenido el ambiente en su aprendizaje y se consultó el criterio de 30 expertos.

En ambas carreras se realizó la comparación de los resultados de las encuestas entre el grupo de estudiantes que recibieron entrenamiento con VIA-ED y el grupo al que se les mostraron sus facilidades, lo que permitió comprobar que existe semejanza de criterios en relación con la repercusión del ambiente, la importancia de VisualProg (Tabla 1) y la posibilidad de generalización de esta experiencia a otras asignaturas o áreas de conocimiento.

Tabla 1. Comparación de la aceptación del sistema VisualProg entre los grupos (obtenido del SPSS)

		Grupo		
		Con entrenamiento en VIA-ED	*Sin entrenamiento en VIA-ED*	*Total*
Recursos: Sistema VisualProg	*Sí Cantidad*	31	32	63
	% del grupo	75.6%	68.1%	71.6%
	No Cantidad	10	15	25
	% del grupo	24.4%	31.9%	28.4%
Total	*Cantidad*	41	47	88
	% del grupo	100%	100%	100%

Sig. del test exacto de Fisher = .434

La integración de los recursos al ambiente constituye un aspecto importante, los mapas conceptuales, aún sin los recursos, ya brindan información, conforman un modelo de conocimiento que el estudiante debe integrar a su estructura cognitiva, los textos, problemas, imágenes, los recursos interactivos, unidos en un mismo ambiente conforman la herramienta que ellos valoran como eficaz. Sin embargo, a los estudiantes sin entrenamiento les es más difícil captar la importancia y el valor que adquiere el sistema que integra estas técnicas.

Al evaluar el SVP VisualProg los estudiantes consideraron que seguir la traza de ejecución y visualización del programa les ayudaba o podía ayudar al diseño eficiente de las estructuras de datos, hubo una tendencia clara hacia las respuestas más positivas. El 88,6% de los estudiantes señalaron que les había permitido comprender bastante o totalmente los contenidos relacionados con el tema, este porcentaje no se diferenció significativamente entre los grupos. La tendencia ascendente es similar desde el punto de vista de los rangos medios de las respuestas, como se ratifica con el test de Mann-Whitney.

La no existencia de diferencia significativa al evaluar esta variable demuestra que para ambos grupos resulta muy significativo un sistema que permite mostrar procesos que de otra forma sería imposible visualizar, como son la representación de las estructuras de datos, variables, bloques de memoria, el árbol de llamadas a funciones y la determinación de la complejidad del programa.

Predominaron, en general, las altas valoraciones sobre el hecho de que VIA-ED se vincule con conceptos estudiados en otras asignaturas y su repercusión en la formación profesional. Un 33% afirmó que fue significativamente positivo y un 63,6% que fue positivo. Incluso este último porcentaje es mayor en el grupo sin entrenamiento, lo que se corrobora con el test de Mann-Whitney.

Uno de los problemas que da origen a la presente investigación es el escaso dominio que tienen los estudiantes de los contenidos previos, necesarios para comenzar a estudiar los temas de Estructuras de Datos. El resultado de la encuesta en esta pregunta muestra la importancia que le confieren los estudiantes de la carrera de Ingeniería Informática e Ingeniería en Sistemas Informáticos, de Cuba y Ecuador, respectivamente (tanto los que usaron VIA-ED, como los que apreciaron esta ventaja, aún sin usarla) a un ambiente que desde su primera interfaz muestra los conceptos objeto de estudio relacionados con los contenidos de asignaturas precedentes. Ello les permite profundizar, consultar, ejercitar y autoevaluarse en los contenidos que consideren necesarios a través de los recursos insertados en estos nodos.

Los criterios sobre la extensibilidad de la experiencia a otros conceptos de Programación y a otros temas de diferentes disciplinas de la carrera fueron ligeramente diferentes entre los grupos, por cuanto, en el caso de los estudiantes sin entrenamiento no se manifestaron dudas al respecto. En cualquier caso, la mayoría se pronunció afirmativamente, de forma condicional (variables: no en todos los casos o sin dudas). Desde el punto de vista de los rangos medios de las opiniones no hay diferencias significativas, como puede apreciarse del test de Mann-Whitney que tiene una significación de 1,0; lo que evidencia que ambos grupos le confieren significación a la propuesta y consideran que puede ser generalizada a otras áreas del conocimiento (Tabla 2).

Tabla 2. Comparación entre los grupos de la extensibilidad de la concepción del ambiente VIA-ED a otras áreas del conocimiento (Obtenido del SPSS)

	Grupo	N	Rango medio	Suma de rangos
Extensibilidad	*Con entrenamiento en VIA_ED*	41	44.51	1825.00
	Sin entrenamiento previo en VIA-ED	47	44.49	2091.00
	Total	88		

Sig. exacta del test de Mann-Whitney = 1.000

En ambos grupos predominan los criterios positivos sobre cómo la forma de representar el contenido que ofrece VIA-ED facilita o puede facilitar la comprensión de los conceptos claves de la asignatura Estructura de Datos. Predominaron en ambos grupos las respuestas positivas (28,4% respondieron que probablemente sí y 64,8% que definitivamente sí) y estos porcentajes son bastante similares entre los grupos.

Por su parte, los expertos de los dos países (Cuba y Ecuador) plantearon satisfacción con el ambiente, sus recursos y técnicas, lo que resalta el valor metodológico para lograr una motivación adecuada de los estudiantes, permite la adquisición de conocimientos y habilidades profesionales mediante su participación activa en el proceso de enseñanza-aprendizaje, a través de la presentación de situaciones problémicas y la interacción con VIA-ED que agrupa los principales contenidos objeto de estudio de la asignatura. Consideran, además, que la experiencia puede ser extendida a cualquier otra materia de la especialidad. La validación de estos criterios se obtuvo a través de la aplicación del método Delphi.

Los resultados obtenidos concuerdan con lo planteado por Almeida, Blanco y Moreno (2011) los que desarrollaron un sitio web sobre estructuras de datos con animación de sus operaciones básicas. Varios autores usan los mapas conceptuales como interfaz de ambientes o plataformas de enseñanza, por las ventajas que ofrecen de organizar conceptos y mostrar relaciones, como es el Mapa Conceptual Hipermedial (MCH) Kellu, creado por Uviña y otros (2015), que muestra las definiciones básicas de trabajo con las estructuras, fundamentalmente, los nexos entre los conceptos, aunque sin apoyarlos con informaciones y recursos.

Moroni y Señas (2015a), plantean que los mapas conceptuales hipermedia constituyen una nueva forma de visualización de programas basada en la representación del código y de la estructura estática del mismo por medio de mapas conceptuales. Además, se apoyan en los resultados de la aplicación del Sistema de Visualización de Programas con Mapas Conceptuales Hipermediales.

La propuesta presentada en este artículo, asume las experiencias en la creación de ambientes de aprendizaje mediados por tecnologías, fundamentalmente con el uso de los mapas conceptuales y las TVP, desarrolla una arquitectura que facilita el desarrollo de sistemas que apoyen a los programadores noveles en la programación de computadoras. El

ambiente VIA-ED, contiene aplicaciones, recursos de información y comunicación que contribuyen a la comprensión y diseño de programas, debido especialmente al sistema VisualProg que, basado en Arq_VP, ayuda al diseño de estructuras de datos y al análisis de la complejidad de programas escritos en el lenguaje SubC.

REFERENCIAS

Almeida, F., Blanco, V. y Moreno, L. (2011). *EDApplets: Una Herramienta Web para la Enseñanza de Estructuras de Datos y Técnicas Algorítmitcas.* Artículo presentado en X Jornadas de Enseñanza Universitaria de la Informática, Universidad de Laguna. Tenerife.

Baecker, R. (2011). *Sorting out Sorting. 16mm color, sound film, 25 minutes. Dynamics Graphics Project.* Artículo presentado en ACM SIGGRAPH'11, Computer Systems Research Institute, University of Toronto, Toronto, Ontario, Canadá.

Bennedsen, J. y Caspersen, M. (2013). Failure rates in introductory programming. *ACM SIGCSE Bulletin, 39*(2), 6.

Bladek, C. y Deek, F. P. (2013). Understanding novice programmers difficulties as a requirement to specifying effective learning environments. *New directions in higher education, Nova Science,* 5.

Brown, M. H. y Sedgewick, R. (2014). *A System for Algorithm Animation.* Artículo presentado en Computer Graphics, Minneapolis, Minn.

Cañas, A. y otros (2009). The network architecture of CmapTools. En *Technical Report IHMC CmapTools*, pp. 11-42. Estados Unidos de América: Institute for Human and Machine Cognition.

Cañas, A. y Novak, J. (2012). *Concept Maps: Theory, Methodology, Technology.* Artículo presentado en Conference on Concept Mapping, Spain.

Chandhok, R. (2010). *Programming Environments based on structure editing: The Gnome approach.* Artículo presentado en National Computer Conference. AFIPS.

Chang, S. y otros (2012). A Visual Language Compiler. *IEEE Transactions on Software Engineering, 21*(1), 506-525.

Chestlevar, C. I. (2015). Utilización de Mapas Conceptuales en la enseñanza de la programación. *Informática Aplicada, 2*, 11.

Clinton, H. (2014). *Program Monitoring and Visualization* (Springer-Verlag ed. Vol. 2). Denver.

Cormen, T. H. y otros (2011). *Introduclion to Algorithms* (2da ed.). McGraw Hill.

Cunningham, W. y Beck, K. (2011). *A Diagram for Object-Oriented Programs.* Artículo presentado en OOPSLA '11, Portland, Oregon.

Deek, F. (2013). *A Visualization Tool Using UML Subset to Aid the Novice Programmer.* Artículo presentado en World Conference on Educational Multimedia, Hypermedia and Telecommunications 2013, Chesapeake, VA: AACE.

Díaz, J. y Leal, P. (2013). *Ambiente Web de Apoyo al Proceso de enseñanza-Aprendizaje a través de la Representación Gráfica de Significados a modo de Mapas Conceptuales*. Barcelona: Paidós.

Eisenstadt, M. y Brayshaw, M. (2011). The Transparent Prolog Machine: an execution model and graphical debugger for logic programming. *Journal of Logic Programming. Human Cognition Research Laboratory Technical Report No. 21a. The Open University. Milton Keynes, MK7 6AA, England*, 9.

Estrada, V. y Febles, J. (2016). *Mapas Conceptuales*. Universidad de Guadalajara, vol. III. México.

Frías, I., Soler, Y. y Lezcano, M. (2014). *Sistema de Enseñanza Asistida por Computadora para la visualización de operaciones sobre estructuras de datos y animación de algoritmos* (tesis de maestría inédita), Universidad de Granma, Bayamo.

George, C. (2010). *EROSI - Visualizing recursion and discovering new errors.* Artículo presentado en ACM SIGCSE Technical Symposium on Computer Science Education.

Graf, M. (2009). A Visual Environment for the Design of Distributed Systems. *Workshop on Visual Languages. IEEE Computer Society, 2*, 330-344.

Guttag, J. y Liskov, B. (1986). *Abstraction and Specification in Program Development.* Leipzig: The MIT Press.

Guzdial, M. y Elliott, A. (2012). *Imagineering inauthentic legitimate peripheral participation: an instructional design approach for motivating computing education.* Artículo presentado en International workshop on Computing education research, Canterbury, United Kingdom.

Hayet, J. B. (2012). *Caminos más cortos.* Artículo presentado en Centro de Investigación en Matemáticas, Guanajuato, México.

Hennessy, S. (2011). Learner perceptions of realism and magic in computer simulations. *British Journal of Educational Technology, 24*.

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

Hyrskykari, A. y Raiha, K.-J. (2007). Animation of Algorithms Without Programming. *IEEE Computer Society. Workshop on Visual Languages*, 40-54.

IBM. (2014). Statistical Package for the Social Sciences (SPSS) (Version 15.0).

Kelly, P. y Keller, M. (2005). Visual Cues: Practical Data Visualization. *IEEE Computer Society Press*, 16.

Liens, L. (2004). Actas del Primer Congreso de Mapas Conceptuales CMC 2004. *CMC 2004.* Recuperado de http://cmc.ihmc.us/CMC2004Programa.html

Liffick, B. y Aiken, R. (2011). *A novice programmer's support environment.* Artículo presentado en Integrating Technology into Computer Science Education.

Lim, D. (2012). Lights, camera, computer science: using films to introduce computer science to non-majors. *Journal of Computing Sciences in Colleges, 23*(5), 58-64.

Moor, B. y Deek, F. (2012). On the Design and Development of a UML-Based Visual Environment for Novice Programmers. College of Computing Sciences, New Jersey

Institute of Technology, Newark, NJ, USA. *Journal of Information Technology Education, 5*, 1-24.

Moriconi, M. y Hare, D. F. (2015). Visualizing Program Designs Through PegaSys. *IEEE Computer, 18(8)*, 72-85.

Moroni, N. y Señas, P. (2015a). Mapas Conceptuales Hipermediales Multidimensionales. *Novatica, 2*(1), 5.

Moroni, N. y Señas, P. (2015b). *SVED: Sistema de Visualización de Algoritmos. Departamento de Ciencias de la Computación*.

Morris, J. (2012). Algorithm Animation: Using algorithm code to drive an animation. *Journal of Visual Langiuages and Computing*, 6.

Moshell, M. y otros (2007). A Spreadsheet-Based Visual Language for Freehand Sketching of Complex Motions. *Workshop on Visual Languages. IEEE Computer Society, 12*(1), 94-104.

Myers, B. (2010). *Incense: A System for Displaying Data Structures.* Artículo presentado en Computer Graphics: SIGGRAPH ’05.

Myers, B. (2011). *Taxonomies of Visual Programming and Program Visualization*. Pittsburgh, PA: School of Computer Science. Carnegie Mellon University.

Naps, T. y Rößling, G. (2014). *Evaluating the Educational Impact of Visualization*: Addison-Wesley Company.

Pane, J. y Myers, B. (2013). Usability issues in the design of novice programming systems (School of Computer Science Technical. Carnegie Mellon University ed., pp. 44). Pittsburgh, PA.

Price, B. A., Baecker, R. M. y Small, I. S. (2014). A Principled Taxonomy of Software Visualization. *Journal of Visual Languages and Computing, 4*(3), 211-266.

Satratzemi, M., Dagdilelis, V. y Evageledis, G. (2010). *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, París, Francia.

Silva, Z. y Hernández, Y. (2013). Acciones metodológicas para el empleo del software educativo como medio de enseñanza. *Opuntia Brava, 2*(5). Recuperado de http://opuntiabrava.ult.edu.cu

Stasko, J. (2010). *TANGO: A Framework and System for Algorithm Animation* (Brown University ed.). Providence, RI 02912.

Stasko, J. (2011). *Software Visualization: Programming as a Multimedia Experience.*: MIT Press.

Stasko, J. (2012). Animating Algorithms with XTANGO. *ACM SIGACT News, 23*, 67-71.

Stasko, J. y Lawrence, A. (2014). Empirically Assessing Algorithm Animations as Learning Aids. *MIT Press*, 25.

Stojanovic, L. (2002). El paradigma constructivista en el diseño de actividades y productos informáticos para ambientes de aprendizaje "on-line". *Pedagogía. Caracas, 23*, 66.

Uviña, P. R. y otros (2015). *Mapas Conceptuales: una herramienta para el aprendizaje de Estructuras de Datos.* Artículo presentado en JEITICS 2015 - Jornadas de Educación en Informática y TICS en Argentina, Dto. Informática-Facultad de Ingeniería-UNPSJB.