HOLOS
ISSN 1807 - 1600

VISUALIZACIÓN DINÁMICA, UNA OPCIÓN PARA LA ENSEÑANZA-APRENDIZAJE DE LA PROGRAMACIÓN DE COMPUTADORAS

Y. S. PELLICER[1], M. C. ZEA[2], R. A. PÉREZ[3], Y. C. BLANCO[4], M. G. L. BRITO[5]
Universidad de Granma[1,4], Universidad Técnica Estatal de Quevedo[2, 3], Universidad Cooperativa de Colombia[5]
yoly@udg.co.cu[1]

Submetido 07/03/2016 - Aceito 28/03/2020

DOI: 10.15628/holos.2020.4241

RESUMEN

La integración de varias Técnicas de Visualización de Programas contribuye a solucionar problemas relacionados con el diseño e implementación de estructuras de datos y programas. En esta investigación se propone el Ambiente Integrado de Visualización de Estructuras de Datos 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 y que integra los componentes de visualización de código, de datos, del árbol de recursividad y complejidad del programa. VisualProg fue implementado teniendo en cuenta una arquitectura concebida en tres capas: Analizador de Código, Controladora y Vista. Se plantean recomendaciones para el uso del ambiente y se exponen los resultados de su evaluación por los expertos y de su aplicación en el proceso de enseñanza-aprendizaje de la asignatura Estructura de Datos de la carrera Ingeniería Informática.

PALABRAS-CLAVE: Sistemas de visualización de programas, Estructuras de Datos, complejidad de algoritmos.

VISUALIZATION OF PROGRAMS, AN OPTION FOR LEARNING COMPUTER PROGRAMMING

ABSTRACT

The integration of various Visualization Techniques Program helps to solve problems related to the design and implementation of data structures and programs. In this research the Integrated Display Data Structures based on concept maps that is a repository of resources, one of which is the VisualProg system that takes as input the code in the language SubC and integrates visualization components Environment proposes code, data, recursion tree and complexity of the program. VisualProg was implemented taking into account an architecture conceived in three layers: Code Analyzer, controller and Vista. recommendations for the use of the environment are posed and the results of evaluation by experts and its implementation are set out in the teaching-learning course Data Structure Engineering career.

KEYWORDS: visualization systems programs, data structure, complex algorithms.

HOLOS
ISSN 1807 - 1600

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 Informática y Ciencia de la Computación. Históricamente, los estudiantes han afrontado problemas para asimilar nociones matemáticas abstractas, en particular cuando éstas incluyen la dinámica de cómo los algoritmos manipulan los datos (Acm e Ieee-Cs, 2009).

La tarea de aprender a manipular el conjunto de símbolos asociado a un lenguaje conforme a una sintaxis, relacionándolos con una semántica, demanda un esfuerzo considerable para los alumnos de los primeros años de las carreras de Ciencias de la Computación. 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.

Reconociendo la situación anterior, muchos centros de enseñanza en todo el mundo, han dedicado grandes esfuerzos a resolverla para lo cual han usado diferentes vías y herramientas que ayudan a mejorar el aprendizaje y el desarrollo de algoritmos, sin embargo la problemática continúa vigente y ello 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 multimedias, con importante interacción entre el usuario y la computadora. La Visualización de Software, por su parte, comprende la de Algoritmos y la de Programas. La primera muestra abstracciones de alto nivel que describen el algoritmo, mientras que la segunda se refiere al código real del programa y a las estructuras de datos. Ambas pueden darse en forma estática o dinámica (Stasko, Domingue et al., 2012).

En este contexto, Stojanovic (2012) y Chestlevar (2015), proponen el uso de Mapas Conceptuales (MC) para la enseñanza de conceptos básicos de programación y desarrollo de algoritmos. Respecto a las destrezas cognitivas, los MC favorecen las conexiones con ideas previas, la capacidad de inclusión, la diferenciación progresiva entre conceptos, la integración, asimilación y muestra de nuevas relaciones entre ellos (Cmc, 2004; Estrada e Febles, 2012).

Moroni e Señas (2006), plantean que los Mapas Conceptuales Hipermediales (MCH) constituyen una nueva forma de visualización de programas, basada en la representación del código y de la estructura estática del mismo

Al revisar los textos tradicionales de enseñanza de la programación de computadoras en el ámbito universitario, se comprueba que mayoritariamente no hacen uso de un lenguaje de diseño de algoritmos para enseñar a programar y, en su lugar, apelan directamente a un lenguaje de programación. Ese acercamiento prescinde, muchas veces, de una clara identificación de cómo se relacionan distintos conceptos teóricos entre sí, el resultado es que muchos se presentan independientemente y sólo a través de la práctica el alumno llega a interrelacionarlos.

A pesar de las facilidades que los lenguajes de programación modernos ofrecen a los programadores, están muy lejos, por ejemplo, de mostrar el efecto que el código escrito provoca en los datos, de visualizar el árbol de llamadas a una función recursiva o de mostrar 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. Según NAPS e 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.

La teoría cognoscitiva de NEWELL (1972) seguida por ANDERSON (1983), KEARNS e VAZIRANI (2004), DEEK e MCHUGH (2008) y BROWN, CARLING *et al.* (2011), recalca el papel determinante que tiene la forma en que el conocimiento, las habilidades y las actitudes se aprenden inicialmente.

En los últimos 30 años las Técnicas de Visualización de Algoritmos y Programas se explotaron y desarrollaron ampliamente y surgieron múltiples sistemas como los desarrollados por MOSHELL, HUGHES *et al.* (1987), CHANG, TAUBER *et al.* (2008) y MAIMONE, TYGAR *et al.* (2009), que usan diversas técnicas de visualización para mostrar el algoritmo, el código, los datos o la estructura de los programas.

Según MYERS (2012a), existen otros Sistemas de Visualización de Programas que se ajustan a múltiples categorías porque muestran diferentes modos de visualización y manejo. Entre estos se encuentran TANGO de STASKO (2005), Diagramas Orientados a Objetos por CUNNINGHAM e BECK (2006) y TPM elaborado por EISENSTADT e BRAYSHAW (2006), Sorting and Sorting desarrollado por BAECKER (2011), PV Prototype diseñado por BROWN, CARLING *et al.* (2011), MacGnome de CHANDHOK (2012), Incense implementado por MYERS (2012b) BALSA de BROWN e SEDGEWICK (2014).

Las instituciones cubanas con carreras de perfil informático, se han preocupado y motivado por la problemática de la enseñanza de la programación y han potenciado investigaciones en colaboración con otras universidades de América Latina que responden a ese campo, lo que ha dado como resultado el diseño e implementación de ambientes de aprendizaje poderosos, que se compenetran con las características de los procesos de aprendizaje en forma efectiva y que involucran una nueva concepción del aprendizaje, entre los que cabe destacar a SESE, SEP y Progen de LEZCANO (1998); LIM y GECOSOFT, desarrollado por SIMÓN, ESTRADA *et al.* (2006) y la organización del conocimiento de la asignatura Estructura de Datos para Ingeniería Informática basada en mapas conceptuales, implementado por SOLER e LEZCANO (2008).

Siguiendo esta línea de investigaciones en este trabajo se presenta el ambiente de Visualización de Programas compuesto por un conjunto de recursos integrados, que permite mejorar el diseño de estructuras de datos y programas y facilita el análisis de eficiencia al mostrar, de forma dinámica, el comportamiento del código, los datos, el árbol de llamadas a funciones recursivas y el análisis de complejidad del programa.

2 VISUALIZACIÓN DINÁMICA DE PROGRAMAS

Se ha experimentado las facilidades que en el campo de la psicopedagogía tiene la Visualización de Software como un medio audio – visual interactivo multimedial. Algunos autores como Guttag e Liskov (1986), Hennessy (2013), Morris (2003), Bladek e Deek (2010), Stasko (2007) y Stasko (2011), consideran que los sistemas no sólo deben estar diseñados por especialistas en computación sino por psicopedagogos que permitan establecer un equilibrio entre el exhibicionismo espectacular que presentan algunas visualizaciones, que a veces no pasan más que por la demostración de los potentes recursos informáticos, y lo que el ser humano puede percibir efectivamente de ellas.

Se han desarrollado numerosos ambientes de programación, aplicaciones y herramientas de enseñanza para ayudar a superar las dificultades afrontadas por los nuevos programadores, sin embargo, investigaciones realizadas en el área de la psicología, la interacción hombre-máquina, la cognición y la pedagogía han identificado y clasificado gran cantidad de problemas que los principiantes afrontan al aprender a programar. Según Bladek e Deek (2010), estas deficiencias pueden ser clasificadas en las categorías siguientes:

2.1 Raíces Pedagógicas:

Una pedagogía ineficaz en la enseñanza de la programación es uno de los factores vitales que influirá en la actuación del programador en experiencias subsecuentes. Liffick e Aiken (2011), observan que al principio los programadores pueden tener dificultades al entender un nuevo concepto, no porque sea difícil, sino porque depende de un concepto anterior. Otro aspecto puede ser atribuido a la inadecuada selección del problema a solucionar para alcanzar determinada habilidad. Suchan e Smith (2007) y Lim (2008), plantean 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 Pane e Myers (2009), así como, Bennedsen e Caspersen (2010), observan que las estrategias de solución al problema general deberían ser explícitamente adquiridas a través de habilidades en el desarrollo de programas.

2.2 Raíces Psicológicas

Una apreciable cantidad de dificultades han sido clasificadas como de raíces psicológicas. 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. Además, algunos de los conceptos de programación, tales como la recursividad, son abstractos por naturaleza, y otros no tienen una contrapartida en el mundo real que facilite la analogía (Guzdial e Elliott, 2012).

También hay que tener en cuenta que, en la mayoría de los casos, los nuevos programadores han sido excelentes usuarios de aplicaciones y en el caso de los estudiantes de Informática y Ciencia de la Computación, a pesar de estar motivados por la carrera, no se han preparado psicológicamente para enfrentar un proceso de creación de aplicaciones y sus dificultades intrínsecas.

Considerando estas bases, Moor e Deek (2013), plantean que las herramientas visuales de aprendizaje pueden ser clasificadas para su análisis en dos categorías, que se corresponden con las actividades de comprensión y construcción:

- Las que emplean la visualización para mostrar y construir animaciones de programas que ya han sido escritos, apoyan la comprensión del código, de conceptos abstractos y el proceso de eliminación de errores. Entre ellas se encuentran sistemas como EROSI (GEORGE, 2010).
- Las que usan la visualización para ayudar a los estudiantes en el proceso de desarrollo de un programa, facilitando el aprendizaje y la modelación de conceptos y procesos de un gran nivel de abstracción, el análisis de programas y la solución de problemas. Estas herramientas constituyen el fundamento teórico de la presente investigación.

Estos empleos de visualización son mutuamente exclusivos y tienen fines completamente diferentes.

3 MATERIALES Y MÉTODOS

Para desarrollar el ambiente, los recursos que lo componen y validar los resultados alcanzados se utilizan diferentes herramientas.

El cmapTools, permite desarrollar el Ambiente Integrado de Visualización de Estructuras de Datos (VIA-ED) que relaciona e integra en un mapa los conceptos de la asignatura Estructura de Datos en la carrera Ingeniería Informática (CAÑAS, HILL *et al.*, 2003; CAÑAS, HILL *et al.*, 2004).

Se diseña 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, desarrollado como parte de la investigación.

Para evaluar el ambiente VIA-ED se realizaron encuestas a estudiantes de la carrera de Ingeniería Informática y se consultó el criterio de expertos cubanos, ecuatorianos y colombianos 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 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 2014-2015.

El segundo grupo lo formaron 47 estudiantes de la Universidad Técnica Estatal de Quevedo, en Ecuador y la Universidad Cooperativa de Colombia 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).

HOLOS
ISSN 1807 - 1600

4 RESULTADOS DE LA INVESTIGACIÓN

El protagonismo de las técnicas de información y las comunicaciones en los ambientes de aprendizajes, ha hecho que el concepto tradicional de software educativo esté migrando, 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 e LEAL, 2012).

En este entorno se presentan los siguientes resultados:

4.1 Ambiente Integrado de Visualización de Estructuras de Datos (VIA-ED)

Teniendo presente la caracterización de las Técnicas de Visualización y las deficiencias encontradas en el diseño e implementación eficiente de estructuras de datos, 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).

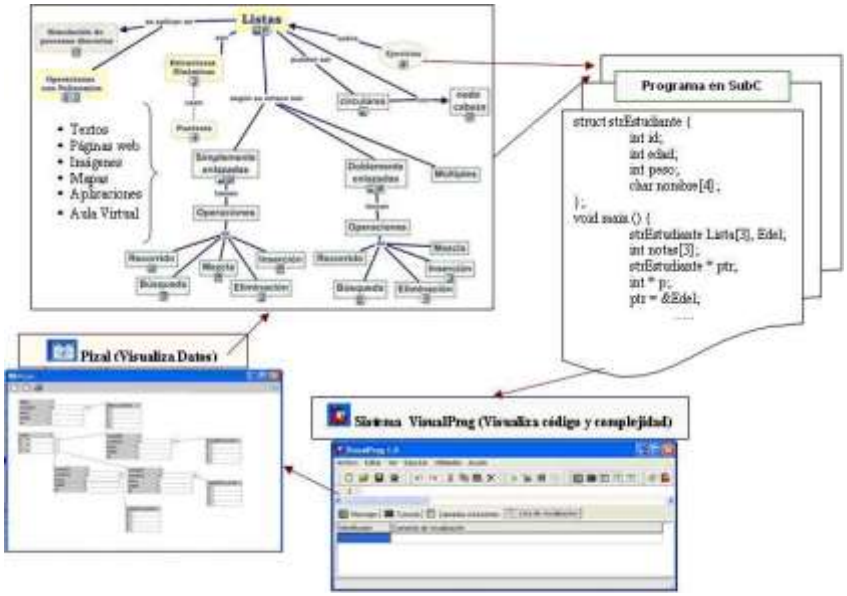


Figura 1. Esquema del Ambiente Integrado de Visualización de Estructuras de Datos

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 cubano, se diseñó el mapa conceptual Tipos Abstractos de Datos (http://cmap.udg.co.cu), 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, usando hilos de discusión sincrónicos y asincrónicos.

4.2 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 usando 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.

4.3 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 que presenta características comunes con las herramientas declarativas e imperativas, ya que en esencia, para llevar a cabo el proceso de visualización en el sistema, se realiza un mapeo de las variables activas en un instante determinado y se representan a objetos de animación que sufrirán cambios en correspondencia con la creación, transformaciones de los valores y destrucción de las variables. Esto se manipulará a través de eventos que sucederán en la ejecución del programa, los que están relacionados con las llamadas a procedimientos y retorno de funciones, paso de parámetros, entre otros. VisualProg está basado en la Arquitectura para desarrollar Sistemas de Visualización de Programas (Arq_VP), desarrollada como parte de la investigación, la que consta de tres capas: Analizador de Código, Controladores y Vistas de código o datos.

VisualProg permite la representación gráfica de los elementos estáticos y dinámicos de programas implementados en un lenguaje de programación semejante al C (SubC), a través de la ejecución real de los mismos.

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.

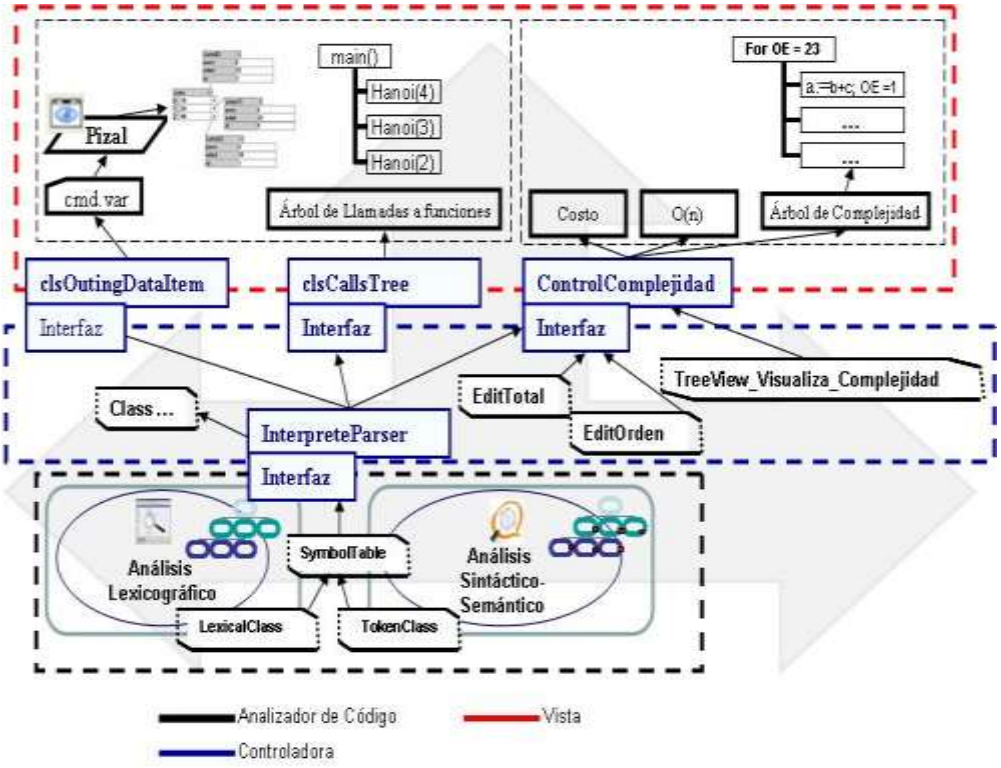


Figura 2. Arquitectura de VisualProg basada en Arq_VP

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.

4.4 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).

HOLOS
ISSN 1807 - 1600

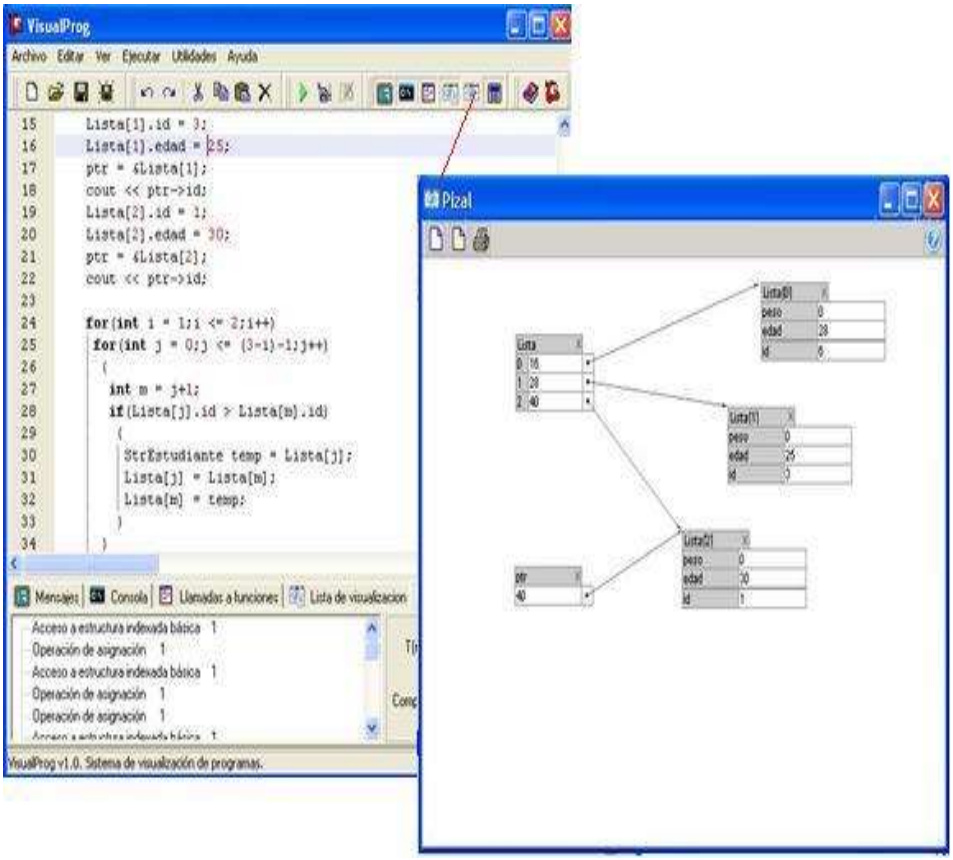


Figura 3. Vista de datos a través de Pizal

En este módulo controlador se usan otros comandos que no están relacionados directamente con los datos pero sí con la gestión de los objetos visuales que los representan como, por ejemplo, eliminar un frame del área de visualización y eliminar todos los frames, dejando limpia el área de trabajo.

4.5 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).

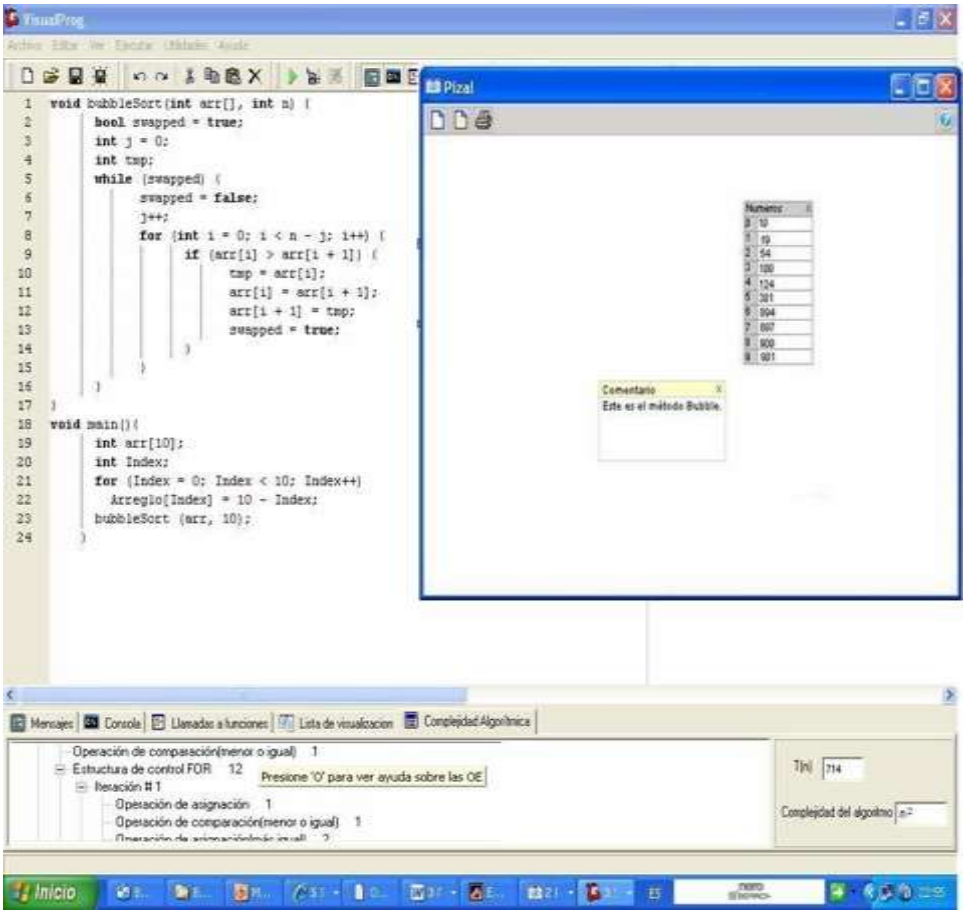


Figura 4. Vista de datos y de la complejidad a través de Pizal.

4.6 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 usando 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 sólo la comprensión, sino también la explicación de estos procedimientos (Figura 5).

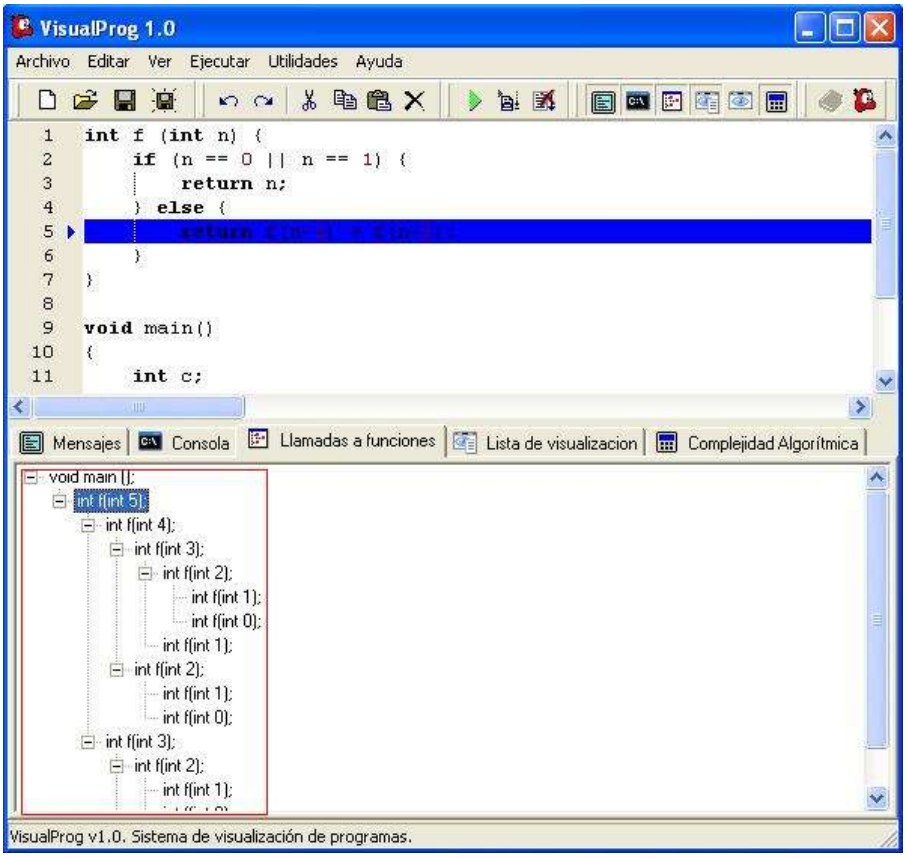


Figura 5. Visualización del árbol de llamadas a funciones

La propuesta del Ambiente Integrado de Visualización de Estructuras de Datos en el proceso de enseñanza-aprendizaje de la asignatura, favorece la actualización de los contenidos 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. A estas cualidades se suman funciones que favorecen el desarrollo del pensamiento y las estrategias cognitivas superiores, por lo que se considera que se ha desarrollado una herramienta adecuado para la práctica educativa. Sin embargo, la simple incorporación de estas tecnologías innovadoras no garantiza la efectividad de los resultados, que se logran, no tanto por el cambio en los medios, sino por la oportunidad que reportan para transformar la filosofía de enseñanza-aprendizaje.

Como resultado se propone un modelo y la metodología para usarlo 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.

4.7 Validación estadística de la efectividad del Ambiente VIA-ED

Para validar la efectividad de VIA-ED se realizaron encuestas a estudiantes de la carrera Ingeniería Informática en las universidades de Granma, Cuba; de Quevedo en Ecuador y la Cooperativa de Colombia, que usaron el ambiente en el proceso de enseñanza-aprendizaje de la asignatura Estructura de Datos en el curso 2013-2014. Se procesaron, además, los criterios de estudiantes que habían recibido la asignatura en cursos anteriores y no tuvieron experiencia de

trabajo con el ambiente, sólo 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. 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 permitió comprobar que existe semejanza de criterios en relación con la repercusión del ambiente, la importancia de VisualProg (Figura 6) y la posibilidad de generalización de esta experiencia a otras asignaturas o áreas de conocimiento.

			Grupo		
			Con entrenamiento en VIA-ED	Sin entrenamiento previo en VIA-ED	Total
Recursos: Sistema de Visualización de programas VisualProg	Si	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	100.0%	100.0%	100.0%

Sig. del test exacto de Fisher=0.434

Figura 6. Comparación de la aceptación del sistema VisualProg entre los grupos (Obtenido del SPSS)

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. En la figura 7 se muestra que un 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.

			Grupo		
			Con entrenamiento en VIA-ED	Sin entrenamiento previo en VIA-ED	Total
Recursos: Sistema de Visualización de programas VisualProg	Si	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	100.0%	100.0%	100.0%

Sig. del test exacto de Fisher=0.434

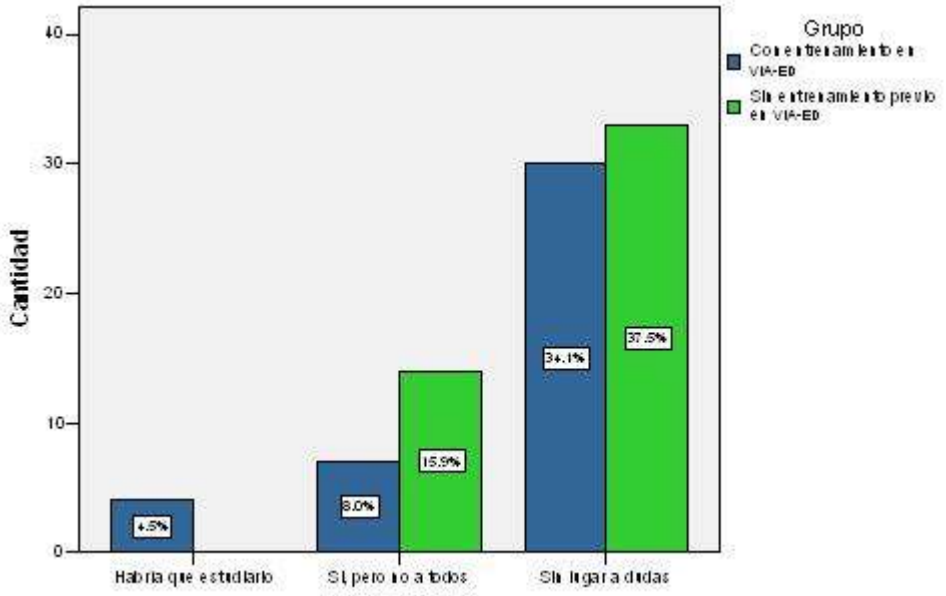
Figura 7. Comparación de la Traza entre los grupos (Test de Mann-Whitney) (Obtenido del SPSS)

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 serían imposibles de 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 altamente positivo. Incluso este último porcentaje es ligeramente mayor en el grupo sin entrenamiento; pero la diferencia es apenas medianamente significativa y efectivamente el test de Mann-Whitney no encuentra diferencias significativas entre los rangos medios.

Uno de los problemas que da origen a la presente investigación está dado por 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 (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, permitiéndoles 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, pues 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 (Figura 8).



Rangos medios según el test de Mann-Whitney

	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

Anexo 8. Comparación entre los grupos de la extensibilidad de la concepción del ambiente VIA-ED a otras áreas del conocimiento (Obtenido del SPSS).

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 último, se compara la respuesta a la pregunta sobre el impacto del ambiente VIA-ED y las Técnicas de Visualización de Programas en la capacidad de los estudiantes para diseñar estructuras de datos y programas eficientes. Predominan las respuestas positivas: apreciable y sobresaliente que alcanzan casi el 90% de la muestra y en ambos grupos de manera similar. El test de Mann-Whitney no encuentra diferencias significativas en los rangos medios.

En el caso de las tres últimas variables analizadas (extensibilidad, representación e impacto), no existen 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.

Por su parte, los expertos de los tres países plantearon satisfacción con el ambiente, sus recursos y técnicas, resaltando el valor metodológico para lograr una motivación adecuada de los estudiantes, permitiendo 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 integra los principales contenidos objeto de estudio de la asignatura. Consideran, además, que la experiencia puede ser extendida a cualquier otra materia de la especialidad.

5 ANÁLISIS Y DISCUSIÓN DE LOS RESULTADOS

Los resultados obtenidos permiten corroborar la importancia de los Sistemas de Visualización de Programas para la enseñanza de la programación, tal y como plantean KELLY e KELLER (2005), PRICE, BAECKER *et al.* (2014) y SATRATZEMI, DAGDILELIS *et al.* (2014). En la presente investigación se integran diferentes Técnicas de Visualización de Programas y se comprueba su impacto en 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 los programas, las estructuras de datos y sus operaciones.

Se coincide con NOVAK e GOWIN (1988) cuando conciben el aprendizaje como procesamiento de información e introducen el Mapa Conceptual como una respuesta a la línea de AUSUBEL, NOVAK *et al.* (2000), del aprendizaje significativo dentro del marco de un programa denominado "Aprender a Aprender". En ellos, el conocimiento está organizado y representado en todos los niveles de abstracción, situando los más generales e inclusivos en la parte superior y los más específicos y menos inclusivos en la parte inferior.

Los materiales elaborados utilizando Mapas Conceptuales facilitan el estudio independiente, permiten que el alumno pueda explorar sus conocimientos previos acerca de un nuevo tema, a la vez que integran la nueva información que ha aprendido, organizan los conocimientos a partir de las principales relaciones entre los conceptos y favorecen el trabajo colaborativo.

Se coincide con ONTORIA (2012), cuando considera que los mapas conceptuales constituyen un recurso esquemático para representar un conjunto de significados conceptuales incluidos en una estructura de proposiciones. Estas pueden ser explícitas o implícitas.

Al confrontar nuestros resultados y los de (ALMEIDA & BLANCO *et al.* (2013)) se corrobora que la visualización de software y especialmente la visualización de programas, contribuye favorable y efectivamente en la comprensión de los mismos. Ayuda tanto en la tarea de interpretación del programa subyacente como en la de diseño, depuración, prueba y mantenimiento del software. Para ello se debe disponer de sistemas que faciliten la interacción hombre – máquina, en cuyo diseño se debe poner especial interés en la percepción humana.

6 CONCLUSIONES FINALES

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 los programas, 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 Sistemas de Visualización de Programas, 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), facilitando la inclusión o modificación de interfaces y procesos al Sistema de Visualización sin que se afecte el sistema.

El Sistema de Visualización dinámico de Programas 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.

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 2013-2014 y fue evaluado satisfactoriamente por estudiantes y expertos.

AGRADECIMIENTOS

Al Centro de Estudios Informáticos de la Universidad Central “Marta Abreu” de Las Villas, a los profesores de la carrera Informática de la Universidad de Granma (Cuba), la Universidad Técnica Estatal de Quevedo (Ecuador), la Universidad Cooperativa de Colombia y, de manera especial, a todos los estudiantes de la carrera Ingeniería Informática por la motivación que dieron a la realización de esta investigación.

REFERENCIAS BIBLIOGRÁFICAS

ACM; IEEE-CS. (2009). Computing Curricula 2009. The Overview Report covering undergraduate degree programs in Computer Engineering, Computer Science, Information Systems Information Technology, Software Engineering. ACM & IEEE-CS, p.58.

BLADEK, C.; DEEK, F. P. (2010). Understanding novice programmers difficulties as a requirement to specifying effective learning environments. New directions in higher education. Nova Science: 5 p.

CHESTLEVAR, C. I. (2015). Utilización de Mapas Conceptuales en la enseñanza de la programación. Informática Aplicada, Bahía Blanca - Argentina, v. 2, p. 11.

CMC. (2004). Actas del Primer Congreso de Mapas Conceptuales CMC 2004. CMC 2004, Universidad Pública de Navarra. Disponível em: < http://cmc.ihmc.us/CMC2004Programa.html >. Acesso em: 3-3-2009.

ESTRADA, V.; FEBLES, J. (2012). Mapas Conceptuales. Universidad de Guadalajara. México.

MORONI, N.; SEÑAS, P. (2006). Mapas Conceptuales Hipermediales Multidimensionales. Novatica, v. 2, n. 1, p. 5.

NAPS, T.; RÖßLING, G. (2014). Evaluating the Educational Impact of Visualization. Addison-Wesley Company. 124-136

NEWELL, A. (1972). Human problem solving. Englewood Cliffs, NJ: Prentice Hall. 49

ANDERSON, J. (1983). The architecture of cognition. 2da. Cambridge, MA: Harvard University Press. 78

KEARNS, M.; VAZIRANI, U. (2004). An Introduction to Computational Learning Theory. The Bactra Review, v. 3, p. 32-56. Disponível em: < http://www.santafe.edu/~shalizi/reviews/kearns-vazirani/ >.

DEEK, F. P.; MCHUGH, J. (2008). An empirical evaluation of specification oriented language in visual environment for instruction translation (SOLVEIT): A problem-solving and program development environment. Journal of Interactive Learning Research, v. 13(4), p. 339-373.

BROWN, G. et al. (2011). Program Visualization: Graphical Support for Software Development. IEEE Computer, v. 18, n. 8, p. 27-35.

MOSHELL, M. et al. (1987). A Spreadsheet-Based Visual Language for Freehand Sketching of Complex Motions. Workshop on Visual Languages. IEEE Computer Society, Linkoping, Sweden, v. 1, p. 94-104.

CHANG, S. et al. (2008). A Visual Language Compiler. IEEE Transactions on Software Engineering, p. 506-525.

MAIMONE, M. W.; TYGAR, J.; WING, J. (2009). Miro Semantics for Security. IEEE Workshop on Visual Languages, Pittsburgh, PA., v. 3, p. 45-51.

MYERS, B. (2012a). Taxonomies of Visual Programming and Program Visualization. Pittsburgh, PA: School of Computer Science. Carnegie Mellon University. 33

STASKO, J. (2005). TANGO: A Framework and System for Algorithm Animation. Brown University. Providence, RI 02912.

CUNNINGHAM, W.; BECK, K. (2006). A Diagram for Object-Oriented Programs. OOPSLA '06 Portland, Oregon: SIGPLAN Notices. 361-367 p.

EISENSTADT, M.; BRAYSHAW, M. (2006). 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, p. 9.

BAECKER, R. (2011). Sorting out Sorting. 16mm color, sound film, 25 minutes. Dynamics Graphics Project. ACM SIGGRAPH'01. Computer Systems Research Institute, University of Toronto, Toronto, Ontario, Canada. 12 p.

CHANDHOK, R. (2012). Programming Environments based on structure editing: The Gnome approach. National Computer Conference. AFIPS. 9-12 p.

MYERS, B. (2012b). Incense: A System for Displaying Data Structures. Computer Graphics: SIGGRAPH '05,. 115-125 p.

BROWN, M. H.; SEDGEWICK, R. (2014). A System for Algorithm Animation. Computer Graphics. Minneapolis, Minn. 177-186 p.

LEZCANO, M. (1998). "Ambientes de aprendizaje por descubrimiento para la disciplina Inteligencia Artificial". 123 Doctorado en Computación y Automática Computación. Facultad de Cibernética-Matemática, Universidad Central "Marta Abreu" de Las Villas, Santa Clara.

SIMÓN, A. et al. (2006). GECOSOFT: un entorno colaborativo para la gestión del conocimiento con mapas conceptuales. Second Int. Conference on Concept Mapping. San José, Costa Rica: A. J. Cañas, J. D. Novak. 4 p.

SOLER, Y.; LEZCANO, G. (2008). Organization of the knowledge of the subjet "Data Structure and algorithms" for informatics ingeneering based on conceptual maps. Revista Avances en Sistemas e Informática, v. 5 n. 3, p. 6.

STASKO, J. et al. (2012). Software Visualization: Programming as a Multimedia Experience. MIT Press,.

STOJANOVIC, L. (2012). El paradigma constructivista en el diseño de actividades y productos informáticos para ambientes de aprendizaje "on-line". Pedagogía. Carácas, v. 23, p. 66.

GUTTAG, J.; LISKOV, B. (1986). Abstraction and Specification in Program Development. Leipzig: The MIT Press.

HENNESSY, S. (2013). Learner perceptions of realism and magic in computer simulations. British Journal of Educational Technology, v. 24.

MORRIS, J. (2003). Algorithm Animation: Using algorithm code to drive an animation. Journal of Visual Langiuages and Computing, p. 6.

STASKO, J. (2007). POLKAW Animation Designer's Package.

STASKO, J. (2011). Animating Algorithms with XTANGO. ACM SIGACT News, v. 23, p. 67-71.

LIFFICK, B.; AIKEN, R. (2011). A novice programmer's support environment. Integrating Technology into Computer Science Education. 49-55 p.

SUCHAN, W.; SMITH, T. (2007). Using Ada as a tool to teach problem solving to non-CS majors. Annual International Conference on Ada. 13 p.

LIM, D. (2008). Lights, camera, computer science: using films to introduce computer science to non-majors. Journal of Computing Sciences in Colleges, v. 23, n. 5, p. 58-64.

PANE, J.; MYERS, B. (2009). Usability issues in the design of novice programming systems. Pittsburgh, PA, p.44. (Report CMU-CS-96-132)

BENNEDSEN, J.; CASPERSEN, M. (2010). Failure rates in introductory programming. ACM SIGCSE Bulletin, v. 39, n. 2, p. 6.

GUZDIAL, M.; ELLIOTT, A. (2012). Imagineering inauthentic legitimate peripheral participation: an instructional design approach for motivating computing education. International workshop on Computing education research. Canterbury, United Kingdom: Spring. 124 p.

MOOR, B.; DEEK, F. (2013). 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, v. 5, p. 1-24.

GEORGE, C. (2010). EROSI - Visualizing recursion and discovering new errors. ACM SIGCSE Technical Symposium on Computer Science Education. 305-309 p.

CAÑAS, A.; HILL, G.; LOTT, J. (2003). Support for constructing knowledge models in CmapTools Institute for Human and Machine Cognition. Pensacola, Fl. (Technical Report. HMC CmapTools 2003-02)

CAÑAS, A. et al. (2004). CMapTools: A Knowledge Modeling and Sharing Enviroment. Disponível em: < http://cmc.ihmc.us/papers/cmc2004-283.pdf >. Acesso em: 20-1-2015.

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

DÍAZ, J.; LEAL, P. (2012). Ambiente Web de Apoyo al Proceso de enseñanza-Aprendizaje a través de la Representación Gráfica de Significados a modo de Mapas Conceptuales. Barcelona: Paidós.

KELLY, P.; KELLER, M. (2005). Visual Cues: Practical Data Visualization. IEEE Computer Society Press, p. 16.

PRICE, B. A.; BAECKER, R. M.; SMALL, I. S. (2014). A Principled Taxonomy of Software Visualization. Journal of Visual Languages and Computing v. 4 (3), p. 211-266.

SATRATZEMI, M.; DAGDILELIS, V.; EVAGELEDIS, G. (2014). A system for program visualization and problem-solving path assessment of novice programmers. Annual Joint Conference Integrating Technology into Computer Science Education, 6th Annual Conference on Innovation and Technology in Computer Science Education.. 137-140 p.

NOVAK, J.; GOWIN, D. (1988). Aprendiendo a aprender. Martínez Roca. Barcelona.

AUSUBEL, D.; NOVAK, J.; HAINESIAN, H. (2000). Psicología Educativa. Un punto de vista cognocitivo. Trillas. México. Disponível em: < F:\Yolanda\ACUÑA CAICEDO ROBERTO.doc >.

ONTORIA, A. (2012). Mapas conceptuales: una técnica para aprender. Narcea S.A. Javeriana.

ALMEIDA, F.; BLANCO, V.; MORENO, L. (2013). EDApplets: Una Herramienta Web para la Enseñanza de Estructuras de Datos y Técnicas Algorítmitcas. X Jornadas de Enseñanza Universitaria de la Informática. Universidad de Laguna. Tenerife. 12 p.