Herramienta para la Visualización de Programas en Lenguajes Imperativos

Norma Moroni – Perla Señas
[nem/ips]@cs.uns.edu.ar

Laboratorio de Investigación y Desarrollo en Informática y Educación (LIDInE)
Instituto de Investigación en Ciencias y Tecnología Informática (IICTI)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur - Bahía Blanca

IX Congreso Argentino de Ciencias de la Computación

Resumen

Se presenta en este trabajo un sistema para la visualización automática de programas escritos en Lenguaje Pascal. Apunta exclusivamente a la comprensión rápida de la estructura estática del programa en general, y del esquema referencial en particular. Se trata de un nuevo modelo de visualización de programas que usa los Mapas Conceptuales Hipermediales como herramienta visual para la representación de las ideas. Está pensado, desde el punto de vista educativo para ayudar a los estudiantes a comprender las estructuras de datos, las técnicas de programación y los nuevos lenguajes que aprenden, y desde el punto de vista profesional para la corrección, el mantenimiento y mejoramiento de programas.

Palabras Clave

Informática Educativa – Visualización de Programas - Mapa Conceptual Hipermedial.

1. Introducción

La Visualización de Software es un campo particular de la visualización computacional que consiste en el uso de recursos gráficos, de animación y multimediales con importante interacción entre el usuario y la computadora. Tiene como finalidad facilitar la comprensión y el uso efectivo del software.
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. La animación de algoritmos muestra la conducta del programa en ejecución, mientras que la visualización de código puede incluir algún tipo de mejoramiento de la impresión como indentado o estructura del programa en forma estática o dinámicamente destacando las líneas de código a medida que ellas están siendo ejecutadas [1]. En este trabajo se propone una nueva forma de visualización de programas. La misma está basada en la representación del código y de la estructura estática del mismo por medio de Mapas Conceptuales.
La automatización de la Visualización de Programas, desde este punto de vista, se logra por la aplicación del Sistema de Visualización de Programas mediante Mapas Conceptuales Hipermediales (MCH) [2]. El sistema permite la representación visual del código de un programa escrito en Lenguaje de Programación Pascal y asegura la correctitud de tal representación. La contribución que esta herramienta brinda es la de favorecer la interpretación de la estructura estática

del programa, el estudio de los ambientes referenciales de los subprogramas que lo componen diferenciando entre los ambientes locales, globales y no locales, las relaciones existentes entre los subprogramas en cuanto a "invocadores de" o "invocados por", el estudio de los parámetros y del pasaje de los mismos, la exhibición del texto del programa y de los distintos subprogramas y la incorporación de mensajes explicativos asociados a los conceptos del MCH.
Las técnicas de Visualización de Software, en general, tienen un importante valor educativo. La animación de algoritmos y la visualización de programas ayuda a los estudiantes a comprender los conceptos de software. El sistema presentado en este trabajo, complementa a SVED [3], Sistema de Visualización de Estructuras de Datos, que permite la animación de dichas estructuras mostrando el comportamiento de las mismas durante la ejecución de un Programa ya creado.

2. Visualización de Programas.

La visualización tiene como meta transformar la información en una más significativa, a partir de la cual el observador humano pueda ganar en comprensión. Con el fin de satisfacer las necesidades de la persona que interactúa con las vistas resultantes de la visualización, todo lo informado a través de la misma debe tener en cuenta aspectos de la percepción [4] y del conocimiento humano. Hay una variedad enorme de aportes sensitivos que pueden favorecer la formación de un cuadro mental. Con tal propósito, la visualización debe buscar estructuras, características, anomalías y relaciones entre los datos objeto de la visualización, presentar una visión global cuando se trata de conjuntos grandes y complejos de datos, y detectar las zonas de interés que merecen un análisis cualitativo focalizado.
En general, para comprender un programa y saber qué hace, se presentan dos alternativas: el estudio del código fuente, o la corrida del programa. En la práctica la lectura del código fuente es incómoda y en muchas situaciones impracticable, y la construcción de casos de prueba para explicar la conducta de un programa es una tarea penosa y especulativa. Estas dificultades motivan el desarrollo de programas o sistemas especiales que son usados para ayudar a explicar la conducta de otros programas [5].
La Visualización de Algoritmos y la de Programas son partes de la Visualización de Software. La primera se refiere a la representación de abstracciones de alto nivel que describen el software, mientras que la segunda apunta al código real de programa y a las estructuras de datos. En ambos casos se puede realizar en forma estática o dinámica. La visualización estática de algoritmos está representada generalmente por medio de organigramas que describen la estructura de su especificación, mientras que la dinámica es lo que se llama animación de algoritmos y muestra su comportamiento en tiempo de corrida. La visualización estática de código puede incluir algún tipo de mejoramiento de la impresión como indentado o estructura del programa, mientras que una representación dinámica del mismo puede destacar las líneas de código cuando éstas están siendo ejecutadas. La visualización estática de datos puede exhibir cuadros de datos mientras que una representación dinámica de los mismos puede mostrar cómo los valores de los datos cambian mientras el programa corre [6]. El Sistema SVED está orientado a Estructuras de Datos y a la animación de su conducta durante la ejecución de un programa. SVED permite la representación de un programa particular con la finalidad de facilitar la interpretación de la conducta del programa evitando las modificaciones de código reiteradas que oscurecen la comprensión del mismo.
Los sistemas para comprender programas son usados en una variedad de aplicaciones. Una de ellas es el estudio de la conducta del algoritmo subyacente que resulta de interés para estudiar estructuras de datos, técnicas de programación, nuevos lenguajes, desde el punto de vista educativo como desde el punto de vista de la corrección, el mantenimiento y mejoras de programas escritos por otras personas.

Las herramientas que realizan análisis estático examinan el texto y proveen información sobre el programa que es válida para todas las ejecuciones independientemente de su entrada [7]. Las técnicas de análisis estático emplean editores de sintaxis, optimizadores de código, embellecimiento de la exhibición del código. Estas herramientas facilitan la legibilidad y no pueden explicar la conducta de un programa [8]. Las herramientas que realizan análisis dinámico proveen información acerca de la ejecución de un programa específico sobre un conjunto particular de datos de entrada. Estos programas complementan las herramientas de análisis estático y proveen la información de la ejecución que no puede hacer la herramienta estática, como los detalles de flujo de control de programa, descripción de estructuras de datos internas. Informan cómo el programa se ejecuta, o pueden presentar información en un tiempo posterior después que la ejecución se completa (análisis post mortem). Los analizadores en tiempo de corrida proveen refuerzo inmediato y permiten al usuario orientarlo respecto de la clase y nivel de detalle de la información monitoreada. Las herramientas en tiempo de corrida pueden ser pasivas o interactivas. En un sistema pasivo, la herramienta presenta información al usuario, pero éste tiene poco control sobre la actividad del programa; en un sistema interactivo, el usuario puede tener control externo sobre la información que se está exhibiendo [9].

3. Sistema para la Visualización de Programas en Lenguajes Imperativos basado en MCH

El Sistema para la Visualización de Programas en Lenguajes Imperativos basado en MCH automatiza la Visualización de Programas. Desde este punto de vista, el sistema permite la representación visual del código de un programa escrito en Lenguaje de Programación Pascal y asegura la correctitud de tal representación realizada sobre el modelo de MCH. El sistema es flexible, en el sentido que posibilita la representación de cualquier programa, y es interactivo ya que permite la modificación del programa. Por otra parte, permite la navegación a través del mapa y cuenta con todas las ventajas multimediales, de los MCH integradores y de los MCH Multidimensionales. La contribución que esta herramienta brinda, es la de favorecer la interpretación de la estructura estática del programa, el estudio de los ambientes referenciales de los subprogramas que lo componen y sus apariencias diferenciando entre los ambientes locales, globales y no locales, las relaciones existentes entre los subprogramas en cuanto a “invocadores de” o “invocados por”, el estudio de los parámetros y del pasaje de los mismos, la exhibición del texto del programa y de los distintos subprogramas y la incorporación de mensajes explicativos del concepto a definir.

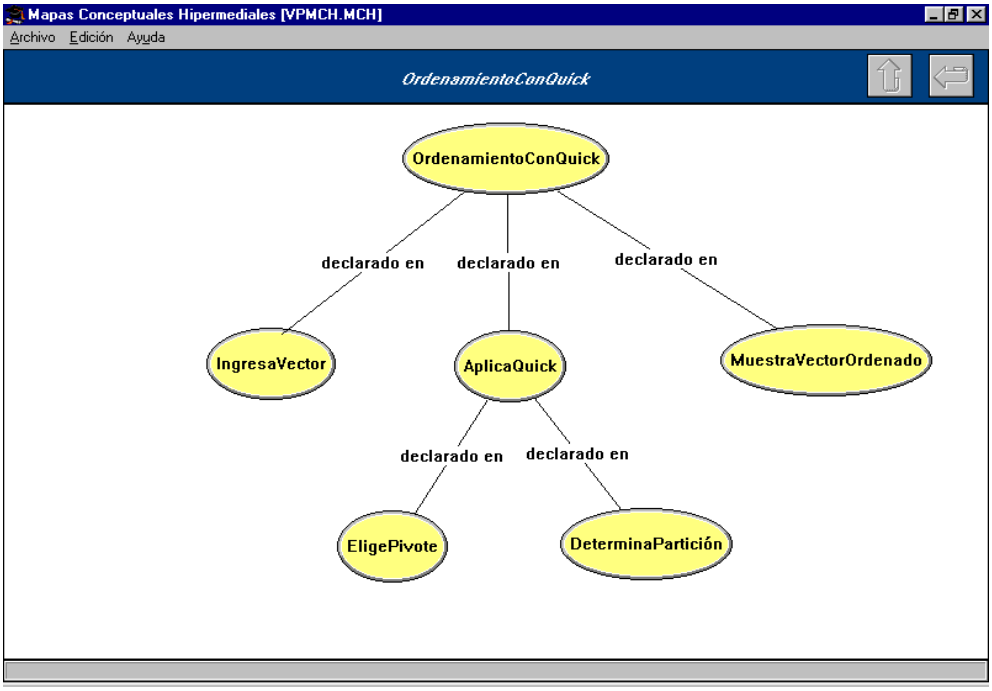
3.1. Ejemplo de Aplicación

A continuación se muestra un ejemplo en el que se realiza la visualización de un programa en Pascal. Sólo se indica la parte del código fuente del programa que interesa para esta aplicación; permite realizar el ordenamiento de un vector empleando el método Quick-sort.

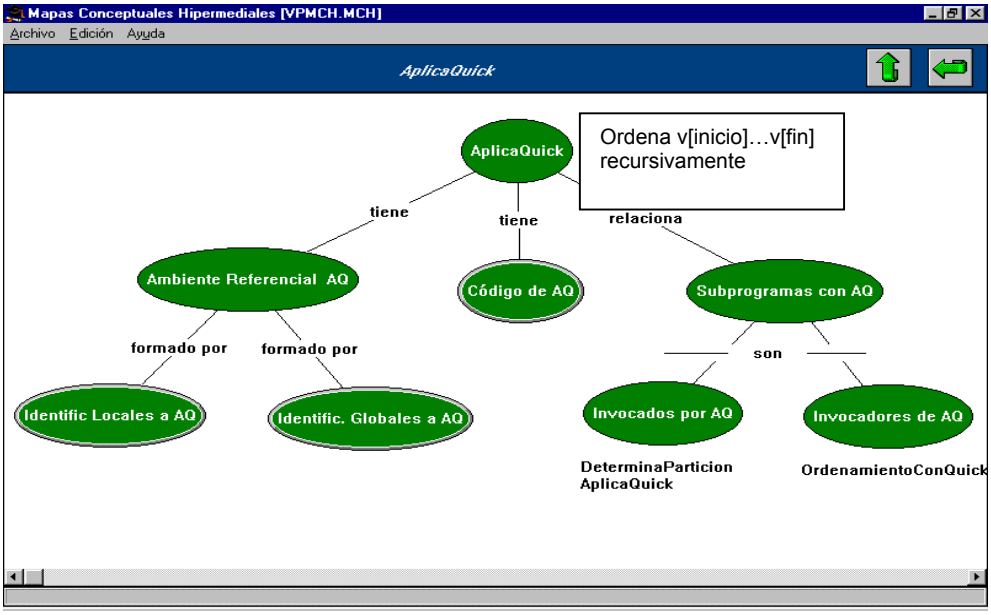
```
Program OrdenamientoConQuick;
{Usa quicksort para ordenar un vector}
  const maximo = 100;

  type          Telemento = Integer;
                Ttvector = array[1..maximo] of Telemento;
      Tindice = 0..maximo;

  var    vector :Tvector;
```

```
        longitud :Tindice;

  procedure IngresaVector ( var a : Tvector; var i : Tindice);
    {permite el ingreso guiado de los valores de los datos}
     var   j :Tindice;
  begin
     ……..
  end;
  procedure AplicaQuick (var v :Tvector ; var  inicio, fin :Tindice);
         {Ordena v[inicio]…v[fin] recursivamente}
          var intermedio :Tindice;

   Function  EligePivote( v: Tvector; inicio, fin :Tindice) : Telemento;
           {Elige el elemento que permite la partición del vector}
     begin
       ……
     end;

  Procedure DeterminaPartición (var v : Tvector; var  inicio, fin, intermedio: Tindice);
           {Reordena v[inicio],…,v[intermedio – 1] <= pivote <= v[intermedio],…,v[fin]}
           var pivote : Telemento;
           begin
             …..
             pivote := EligePivote ( v, inicio, fin);
             …..
           end;

  begin
     ……
    DeterminaPartición( v, inicio, fin, intermedio);
    AplicaQuick (v, inicio, intermedio-1);
    AplicaQuick (v, intermedio, fin);
  end;

  Pocedure MuestraVectorOrdenado ( a :TVector, n :TIndice);
     { Muestra por pantalla el vector ya ordenado}
  var  i: TIndice;
  begin
  …….
  end;

Begin {Quicksort}
  IngresaVector (vector, longitud);
  AplicaQuick (vector, 1, longitud);
  MuestraVectorOrdenado ( vector, longitud)
end.
```

El sistema crea automáticamente el Mapa Conceptual Hipermedial de nombre *VPMCH* que muestra en su primera vista el mapa conceptual que representa la estructura estática del programa y sus subprogramas. El concepto raíz es una elipse asociada al programa, mientras que los restantes conceptos de esta vista inicial son botones cada uno de los cuales está vinculado a un subprograma. Dichos botones permiten la navegación hacia las vistas encargadas de mostrar la estructura del procedimiento/función que representa.

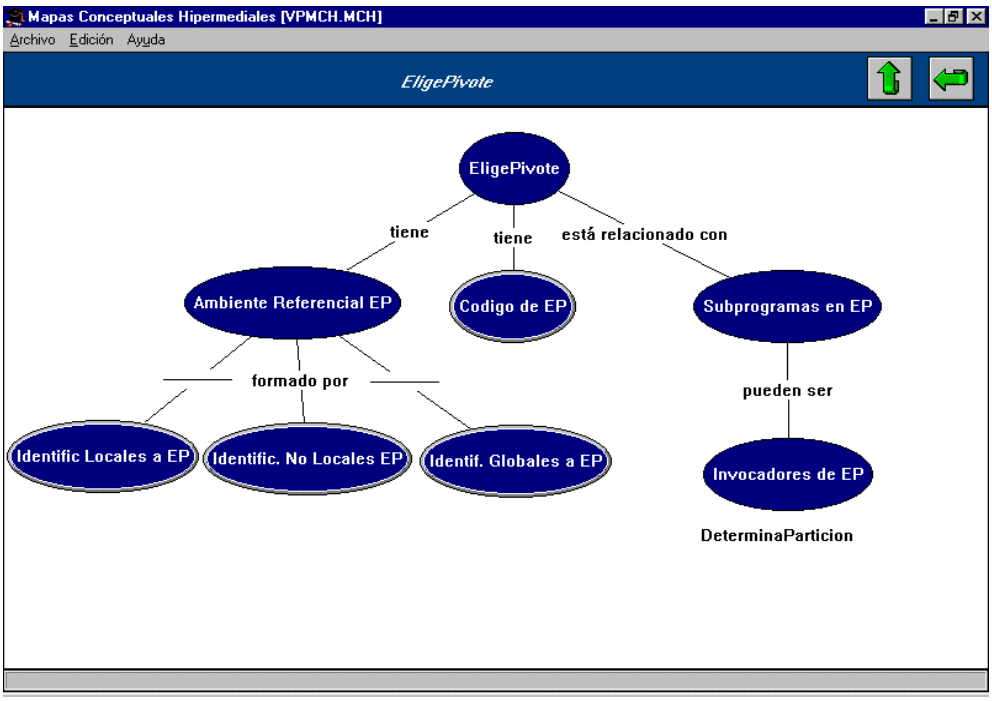


Se puede observar en la vista *OrdenamientoConQuick* que tanto el programa como los subprogramas son conceptos botones. Cada uno de ellos explota en una nueva vista que presenta distintas características del programa o subprograma.

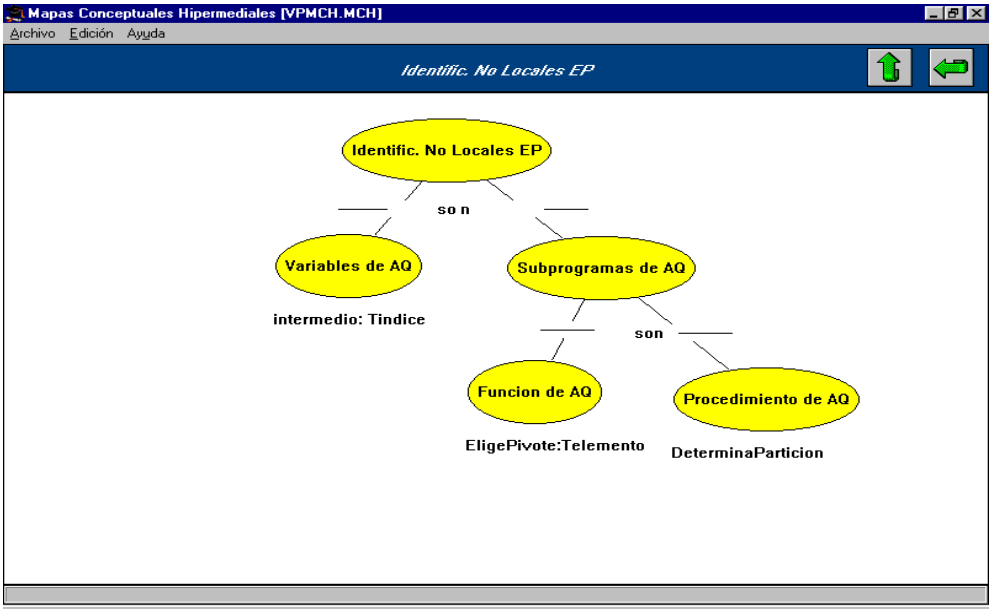


Si se cliquea sobre el concepto *AplicaQuick* se obtiene la vista que muestra los conceptos de Ambiente *Referencial*, *Texto* y *Subprogramas* con los que está relacionado el concepto raíz. El concepto *AplicaQuick* tiene como el texto del mensaje que explica la tarea que realiza.
Se puede observar que el Ambiente Referencial de AQ está formado sólo por identificadores locales y globales, mientras que el Ambiente Referencial de *EligePivote* también cuenta con identificadores no locales, como se verá más adelante. Por otra parte, el botón al que hicimos referencia presenta una apariencia que consiste en un diagrama de conjuntos que representa el anidamiento de los

procedimientos y el ambiente referencial del subprograma más anidado. Además, se destaca en los subprogramas invocados por AQ el uso de una técnica muy útil de programación como lo es la recursión. El botón *Código de AQ* explota en el texto del procedimiento *AplicaQuick*. Tanto para el programa como para los subprogramas las vistas tienen representaciones similares. No se incluyen los conceptos que quedan vacíos (como se observa en AplicaQuick la ausencia del concepto Identific.No Locales).



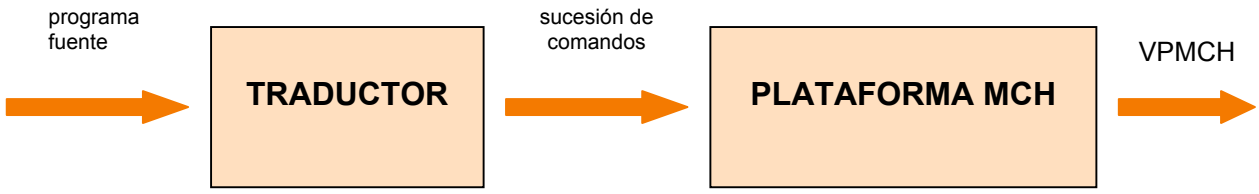
Si se explota el botón *Identific. No Locales a EP* se obtiene la vista que describe cada uno de ellos. Se puede observar que se indica tanto al identificador, al tipo de dato (si corresponde) y al subprograma donde dicho identificador está declarado.



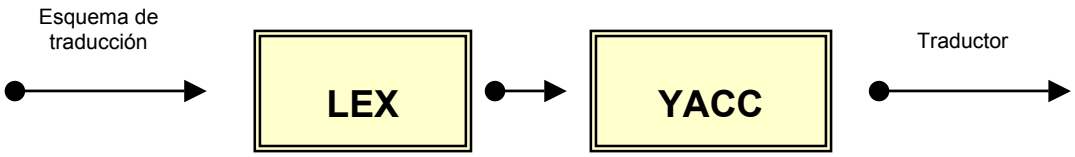
Aplicando el Grafo Integrador para Mapas Conceptuales Hipermediales [10], se puede obtener un único mapa que reúne todos los conceptos que se encuentran distribuidos en las vistas. Con ello se pueden apreciar distintos elementos como, por ejemplo, todo el ambiente referencial de un determinado procedimiento. Por otra parte, aplicando la Plataforma de MCH Multidimensionales [11], podría distinguirse el estudio de los elementos considerándolos desde distintos puntos de vista, por ejemplo, las variables y parámetros asociados a sus tipos de datos.

3.2. Diseño del Sistema

El sistema consiste en un traductor que permite el ingreso de un archivo de texto que contiene el programa fuente en Leguaje Pascal y al que se le desea hacer la visualización. Da como salida el MCH correspondiente.
Posee una primer fase de análisis, léxico, sintáctico y semántico, que permite determinar la validez de las partes declarativas del programa fuente y una segunda fase de generación de código intermedio en formato MCH. Si en la fase de análisis detecta errores, los informa y genera el MCH que remarca las mencionadas situaciones de manera especial.



Una implementación sencilla del traductor para este sistema de visualización puede realizarse mediante el uso de los generadores LEX y YACC. Recibirán como entrada un esquema de traducción basado en las descripciones del Lenguaje Pascal y de los MCH y como salida el programa traductor antes mencionado.



4. Conclusión

El uso de Mapas Conceptuales Hipermediales para la visualización conceptual de un programa potencia las técnicas de visualización aplicadas hasta el momento. Presenta una alternativa novedosa de presentación y tiene la ventaja de realizarse en forma automática. La entrada consiste en el programa o segmentos del texto donde figuren las entidades que intervienen en la representación.
Las técnicas de Visualización de Software, en general, tienen un importante valor educativo. La Animación de Algoritmos y la Visualización de Programas ayudan a los estudiantes a comprender los conceptos de software. El sistema presentado en este trabajo, complementa a SVED que permite la animación de las Estructuras de Datos mostrando el comportamiento de las mismas durante la ejecución de un Programa ya creado.

Este nuevo sistema está pensado, desde el punto de vista educativo para ayudar a los estudiantes a comprender las estructuras de datos, las técnicas de programación y los nuevos lenguajes, y desde el punto de vista profesional para la corrección, el mantenimiento y mejoramiento de programas.

5. Referencias Bibliográficas

[1] Stasko, J., Domingue, J., Brown, M., Price, B. *Software Visualization: Programming as a Multimedia Experience.* MIT Press, 1998.

[2] Moroni, N. - Vitturini, M. - Zanconi, M. - Señas, P. *"Una plataforma para el desarrollo de mapas conceptuales hipermediales"*. IV Jornadas Chilenas de Computación. Valdivia. 1996.

[3] Moroni N. – Señas P. *SVED: Sistema de Visualización de Estructuras de Datos*. CACIC 2002.

[4] Grinstein G - Levkowits H. "*Perceptual Issues in Visualization*", Springer-Verlag, 1995.

[5] Clinton L Jeffery. *Program Monitoring and Visualization*. Springer-Verlag. 1999.

[6] Brown, M. 1992**.** *Zeus: A System for Algorithm Animation and Multi-view Editing***.** Technical report SRC-75, Digital - Systems Research Center.

[7] Price B., Beacker R. y Small I. *An Introduction to Software Visualisation.* Software Visualisation. Cap 27.MIT Press. 1998.

[8] Lawrence, A., Brade, A., Stasko, J.*, Empirically Evaluating the Use of Animations to Teach Algoritms*. Technical Report GIT-GVU-94-07, Graphics, Visualisation, and Ussability Center, College of Computing. Georgia Institute of Technology.

[9] Brown y Sedgewick. *A system for Algorithm Animation*. ACM Computer Graphics. 1985.

[10] Martig, S. y Señas, P. *Grafo Integrador de un MCH*. Enviado a VI WIE. Brasil. 2000.

[11] Moroni, N. y Señas, P. *Mapas Conceptuales Hipermediales Multidimensionales*. VI WIE. Brasil. 2000.

[12] Aho,A., Sethi, R. and Ullman, J., *Compiladores: Principios, Técnicas y Herramientas*. Addison- Wesley. 1986.

.