Concept Maps: Theory, Methodology, Technology
Proc. of the First Int. Conference on Concept Mapping
Pamplona, Spain 2004

APLICACIÓN DE MAPAS CONCEPTUALES HIPERMEDIALES EN LA VISUALIZACIÓN DE PROGRAMAS

Norma Moroni - Perla Señas
Laboratorio de Investigación y Desarrollo en Informática y Educación (LIDInE)
Instituto de Investigación en Ciencias y Tecnología Informática (IICyTI)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur Bahía Blanca - Argentina
[nem@cs.uns.edu.ar / psenas@cs.uns.edu.ar]

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

1 Introducción

La Visualización de información en general consiste en el uso de recursos gráficos, de animación y multimediales con importante interacción entre el usuario y la computadora. Específicamente, la Visualización de Software, que usa recursos similares a la de información, tiene como finalidad facilitar la comprensión y el uso efectivo de programas.

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 destacando las líneas de código a medida que ellas están siendo ejecutadas, dinámicamente (Stasko y otros, 2000).

En este trabajo se propone una nueva forma de visualización de programas. Ésta está basada en la representación del código y de la estructura estática del mismo por medio de Mapas Conceptuales. Por medio del Sistema de Visualización de Programas con Mapas Conceptuales Hipermediales (VP_{MCH}) se logra la automatización de la Visualización de Programas. 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, no locales y globales, 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 ayudan a los estudiantes a comprender los conceptos de software. El sistema presentado en este trabajo, complementa a SVED (Moroni, 2002) Sistema de Visualización de Estructuras de Datos, que permite la animación de las Estructuras de Datos 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 presentaciones resultantes de la visualización, todo lo informado a través de la misma debe tener en cuenta aspectos de la percepción (Grinstein y Levkowits, 1995) 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 (Clinton, 1999).

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 (Price y otros. 1998). Las técnicas de análisis estático emplean editores de sintaxis, optimizadores de código, embellecimiento de la exhibición del código. El uso de mapas conceptuales constituye una novedad para la visualización automática de programas.

3 Sistema para la Visualización de Programas basado en MCH

El Sistema de Visualización de Programas por medio de Mapas Conceptuales Hipermediales (VP_{MCH}) 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 y es interactivo ya que posibilita la representación de cualquier programa y permite la modificación del mismo. 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, no locales y globales, 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. El sistema acepta como entrada un programa imperativo y exhibe el MCH correspondiente. Su uso está orientado fundamentalmente como complemento en la enseñanza de programación y de ambientes referenciales que se estudian en las primeras materias de la carrera de Licenciatura en Ciencias de la Computación y la de Ingeniería en Sistemas de Computación , de la Universidad Nacional del Sur en Argentina.

3.1 Ejemplo de Aplicación

A continuación se muestra en la **Figura 1** 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. El programa describe el ordenamiento de un vector empleando el método Quicksort.

El sistema, una vez ingresado el programa, crea automáticamente el Mapa Conceptual Hipermedial de nombre VP_{MCH} 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. Las relaciones establecidas entre los subprogramas y el programa responden a la ubicación de la declaración de los mismos dentro del programa. Los botones permiten la navegación hacia las vistas encargadas de mostrar la estructura estática del procedimiento o de la función que representan. Estas nuevas vistas también presentan ambientes referenciales, relaciones entre los subprogramas invocados en ellos o que los invocan, destacando como ejemplos los identificadores respectivos.

Se puede observar en la vista *OrdenamientoConQuick*, en la **Error! Reference source not found.**, 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 apariencia el texto del mensaje que explica la tarea que realiza. Se puede observar, en la **Figura 2**, que el Ambiente Referencial de AQ está formado sólo por identificadores locales y globales. 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.

```
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 (
    {Permite el ingreso guiado de los valores de los datos}
    var   j :Tindice;
    begin  ……..   end;
  procedure AplicaQuick (var v :
    {Ordena v[inicio]…v[fin] recursivamente}
    var intermedio :Tindice;
      function  EligePivote( v: Tvecto
      {Elige el elemento que permite la partición d
      begin  ……  end;
      procedure DeterminaPartic
      {Reordena v[inicio],…,v[intermedio – 1] <= pivote <= v[
      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;
  procedure 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.
```

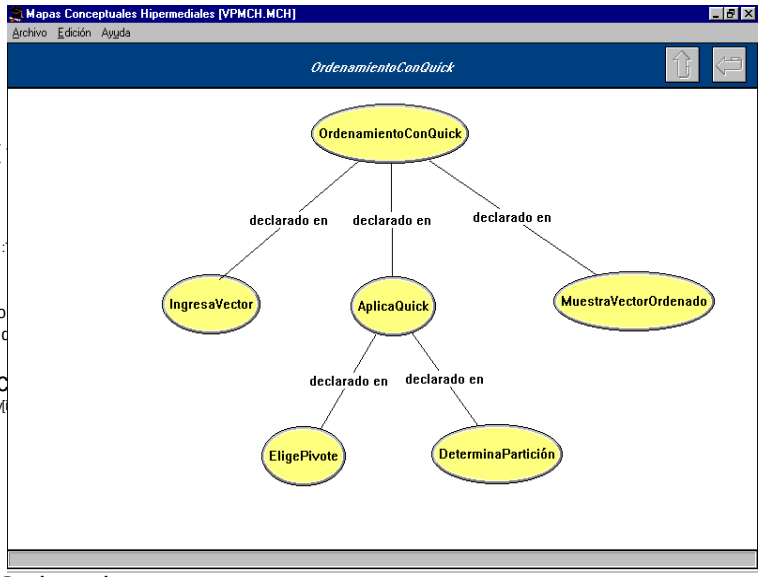

Figura 1

```
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: Tindic
       {Reordena v[inicio],…,v[i
       var pivote : Telemento;
       begin
         ….. pivote := EligePivo
       end;
  begin
    DeterminaPartición( v,
    AplicaQuick(v, inicio, inte
    AplicaQuick (v, intermed
  end;
```

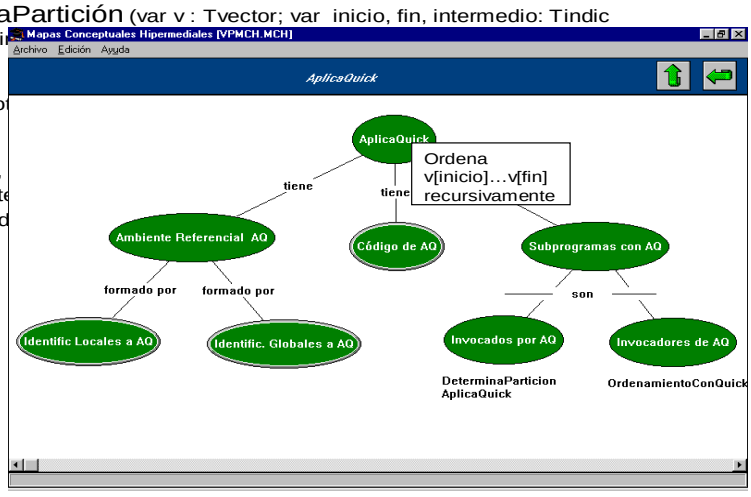

Figura 2

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.

Conclusión

El uso de Mapas Conceptuales Hipermediales para la visualización conceptual de un programa potencia las técnicas de visualización aplicadas hasta el momento. Constituye una alternativa novedosa de presentación y tiene la ventaja de realizarse en forma automática. Este nuevo sistema está pensado, desde el punto de vista educativo para ayudar a los estudiantes a comprender los ambientes referenciales, las técnicas de programación y los nuevos lenguajes, y desde el punto de vista profesional para la corrección, el mantenimiento y el mejoramiento de programas. Además, resulta atractivo como complemento de la documentación de un programa.

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.

Referencias Bibliográficas

Aho, Sethi, y Ullman. (1986). *Compiladores: Principios, Técnicas y Herramientas.* Addison- Wesley..

Brown y Sedgewick. (1985). *A system for Algorithm Animation.* ACM Computer Graphics.

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

Cañas, A. J. *Algunas Ideas sobre la Educación y las Herramientas Computacionales Necesarias para Apoyar su Implementación*, Memoria del IX Congreso Internacional sobre Tecnología y Educación a Distancia, San José, Costa Rica. Reimpreso en Red: Educación y Formación Profesional a Distancia, Ministerio de Educación, España (1999).

Clinton. (1999). Program Monitoring and Visualization. Springer-Verlag.

Gaines, B. and Shaw, M *Open Architecture multimedia documents*. Proceedins ACM Multimedia 93. 1993.

Grinstein y Levkowits. (1995). *Perceptual Issues in Visualization*", Springer-Verlag.

Lawrence, Brade, Stasko. (2000). *Empirically Evaluating the Use of Animations to Teach Algoritms.* Technical Report, Graphics, Visualisation, and Ussability Center, College of Computing. Georgia Institute of Technology.

Martig y Señas. (2000). *Grafo Integrador de un MCH*. Enviado a VI WIE. Brasil.

Moroni y Señas. (2000). Mapas Conceptuales Hipermediales Multidimensionales. VI WIE. Brasil.

Moroni y Señas. (2002). SVED: Sistema de Visualización de Estructuras de Datos. CACIC 2002.

Price, Beacker y Small. (1998). *An Introduction to Software Visualisation.* Software Visualisation. Cap 27. MIT Press.

Stasko, Domingue, Brown, Price. (1998). Software Visualization: Programming as a Multimedia Experience. MIT Press.

Zanconi, M., Moroni, N., Vitturini, M., Malet, A., Borel, C. y Señas, P. *Tecnología computacional y meta-aprendizajes*. RIBIE-98. Brasil. 1998.