DISEÑO, ESPECIFICACION Y PROTOTIPIFICACION DE SISTEMAS INTERACTIVOS

Víctor Manuel Toro C.

Grupo "**ISAC**: Ingeniería de Software Apoyada por Computador"
Departamento de Sistemas y Computación
Universidad de los Andes
Apartado Aéreo 4976
Bogotá - Colombia

"XIII Conferencia Latinoamericana de Informática"
Bogotá, Noviembre'87

RESUMEN: Para describir globalmente un sistema de software es imprescindible disponer de un "lenguaje" capaz de expresar de manera general (¡pero concreta!) los principales aspectos del sistema. La escogencia de este lenguaje es particularmente crucial para la labor de Diseño, puesto que esta etapa tiene justamente el objetivo de describir sistemas de software que aún no existen. Un buen lenguaje de diseño debe ser general, con una clara interpretación intuitiva, y con un sólido respaldo formal. Pero además, debe ser susceptible de procesamiento automático, de manera tal que se pueda avanzar fácilmente hacia la siguiente etapa de desarrollo: la programación. Este artículo presenta una metodología (i.e. herramientas + criterios) clara y efectiva para diseñar sistemas interactivos de complejidad mediana. Se explora asímismo la posibilidad de procesar automáticamente dichos diseños, con miras a obtener directamente prototipos del sistema en construcción.

0. INTRODUCCION

"... hay que terminar rápidamente la etapa de diseño, para que al final del proyecto quede tiempo suficiente para poder resolver los problemas causados por haber hecho muy rápido la etapa de diseño ..." [MYER 78].

La Ingeniería de Software está cambiando radicalmente. El clásico Ciclo de Vida del Software, sobre el cual se apoyan numerosos conceptos y métodos, está dejando de ser adecuado. En su lugar están apareciendo diversas propuestas: prototipificación, especificación operacional, desarrollo transformacional, metodología Jackson, ... [AGRE 86].

Uno de los aspectos más polémicos es el Diseño. En realidad, desde hace bastantes años existen métodos y lenguajes para diseñar (i.e. describir globalmente antes de desarrollar) sistemas de software comercial, que tradicionalmente daban lugar a aplicaciones tipo "batch" [YOUR 76]. Sin embargo, estos métodos y lenguajes de diseño empezaron a mostrarse inadecuados en muchos casos, y se fueron convirtiendo más en un obstáculo que en una ayuda [YOUR 86]. Como consecuencia, en los últimos años han sido identificadas al menos cuatro categorías diferentes de diseño [PRES 87]:

- diseño orientado por flujo de datos
- diseño orientado por objetos
- diseño orientado por estructuras de datos
- diseño de sistemas de tiempo real.

En este artículo se presenta una metodología apropiada para diseñar, especificar y prototipificar sistemas interactivos. Esta metodología, que puede ser clasificada en la línea de diseño orientado por objetos, fué afinada a lo largo de un proyecto culminado recientemente, en el cual se construyó un ambiente de aprendizaje de la programación [LOPE 87a]. Adicionalmente, ha venido siendo probada en varios cursos-proyecto de pregrado y postgrado, en general con muy buenos resultados. En este artículo se tratarán los siguientes temas:

1. Caracterización de los Sistemas Interactivos
2. Los Diagramas de Transiciones
3. Especificación detallada de Sistemas Interactivos.
4. "Algoritmo" para diseñar Sistemas Interactivos
5. Compilación de Diagramas Estructurados de Transiciones
6. Prototipificación de Sistemas Interactivos
7. Conclusiones

Es de anotar que algunos ejemplos y analogías presentados en esta trabajo harán referencia al conocido editor WordStar®. Esto se hace únicamente para facilitar la explicación de ciertos conceptos, pero no implica en ningún momento que lo aquí expuesto solo tenga aplicación a los problemas de edición de texto.

1. CARACTERIZACION DE LOS SISTEMAS INTERACTIVOS

Un Sistema Interactivo S./I. es un conjunto de servicios y facilidades a disposición del usuario, que funcionan, se activan o suspenden siguiendo las órdenes del usuario. Algunas de las diferencias fundamentales entre los Sistemas Interactivos y los sistemas "batch" clásicos son las siguientes:

- Los S./I. no poseen "entradas" y "salidas" definidas de antemano (p. ej.: relación de horas extras, novedades de descuentos ➡ Cheques de nómina, relación de retenciones). En los S./I. la entrada es una sucesión de datos y comandos seleccionados directamente y sobre la marcha por el usuario (p. ej.: letras, números, 'suba renglón', 'borre palabra', 'cambie margen', 'inserte',...) que eventualmente producirán como resultado lo que el usuario quiere obtener.

- Un S./I. no "hace" una determinada labor (p.ej.: procesar la nómina, actualizar el inventario, ...), sino que facilita que el usuario haga su propia labor (p.ej.: preparar un informe, calcular un presupuesto, evaluar un diseño,...).

- Los S./I. no son susceptibles de ser calificados como globalmente correctos o incorrectos. En efecto, la noción de corrección es una relación entre entradas y salidas bien determinadas [GRIE 81][BOHO 86], lo cual no es el caso de los S./I.. La corrección en los sistemas interactivos está confinada a la implementación correcta de cada una de las diversas facilidades y servicios que se ofrecen al usuario.

El objetivo primordial de un S./I. es ser ***adecuado***, es decir, permitir que el usuario pueda realizar efectivamente las funciones que necesita para su labor. En otras palabras, el sistema interactivo debe ofrecer al usuario las facilidades y servicios suficientes para que éste pueda realizar su labor rápida y cómodamente.

Esta es justamente la principal deficiencia de los métodos clásicos de diseño: no considerar ni permitir expresar en forma explícita el papel activo del usuario. Para describir correctamente un sistema interactivo es necesario entonces un lenguaje y un método de diseño en el que:

- aparezca explícito el papel del usuario
- se muestren los servicios que el usuario tiene eventualmente accesibles en cada instante
- se muestre la manera de seleccionar tales servicios
- se pueda especificar rigurosamente la acción que realiza cada uno de dichos servicios

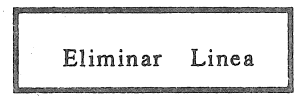
2. LOS DIAGRAMAS DE TRANSICIONES

Para describir Sistemas Interactivos se van a utilizar los "Diagramas de Transiciones". Este formalismo fué sugerido inicialmente por A. I. Wasserman [WASS 81], aunque sin mayor elaboración. En este trabajo se ha refinado dicho formalismo, incorporándole además conceptos modernos como especificación formal [GEHA 86], certificación de la corrección del software [GRIE 81][BOHO 86], y tipos abstractos de datos [CAR86][CAR87].

Un **Diagrama de Transiciones** es un grafo Bi-Partito definido por:

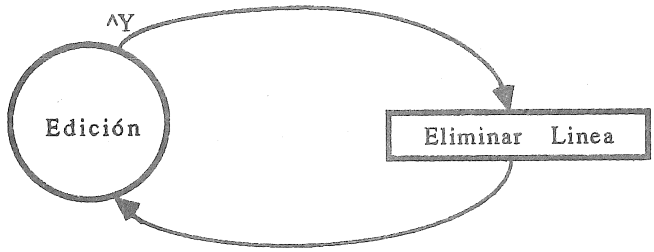
Nodos:

* Conjunto de Niveles p.ej.: 

* Conjunto de Procesadores de Comando p.ej: 

Arcos:

- Los arcos van de un "Nivel" a un "Procesador de Comando", o viceversa.
- Los arcos que van de un Nivel a un Procesador de Comando pueden estar marcados con una etiqueta llamada "Selector".
- Se llaman "Transiciones" a los caminos de la forma siguiente:



ó bien

--- * ---

Los Niveles modelan los *estados de interacción*, esto es, los instantes de un sistema interactivo en los cuales el usuario decide cual es la siguiente acción que debe ejecutarse. Más adelante se verá que, desde un punto de vista conceptual, un Nivel no es más que la materialización de un Tipo Abstracto de Datos [CARD 86][CARD 87], es decir, un objeto más un repertorio de acciones o comandos que pueden actuar sobre él.

Los Procesadores de Comando modelan los *estados de ejecución*, esto es, la ejecución de las acciones que ordena el usuario.

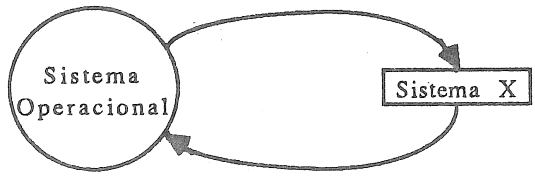
Las Transiciones modelan la secuencia "el usuario puede seleccionar alguna acción", "se realiza la acción seleccionado por el usuario", "el usuario puede seleccionar la siguiente acción".

---*---

De la anterior definición se deduciría que cualquier grafo bi-partito de 'Niveles' y 'Procesadores de Comando' sería un Diagrama de Transiciones válido. Sinembargo, con el fin de poder manipular estos diagramas con herramientas y procedimientos automatizados, es conveniente restringirse a una clase que llamaremos los "Diagramas Estructurados de Transiciones". La práctica muestra que esta restricción no impide expresar complejos sistemas interactivos: todo lo contrario, permite hacerlo con mayor claridad y certeza.

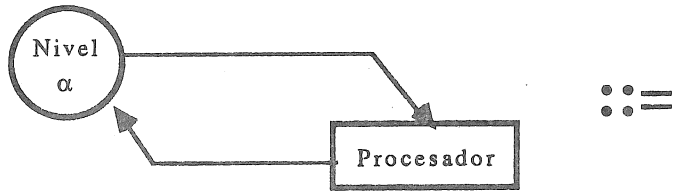
Los **Diagramas Estructurados de Transiciones** son aquellos que se derivan de la siguiente gramática:

Símbolo Distinguido:

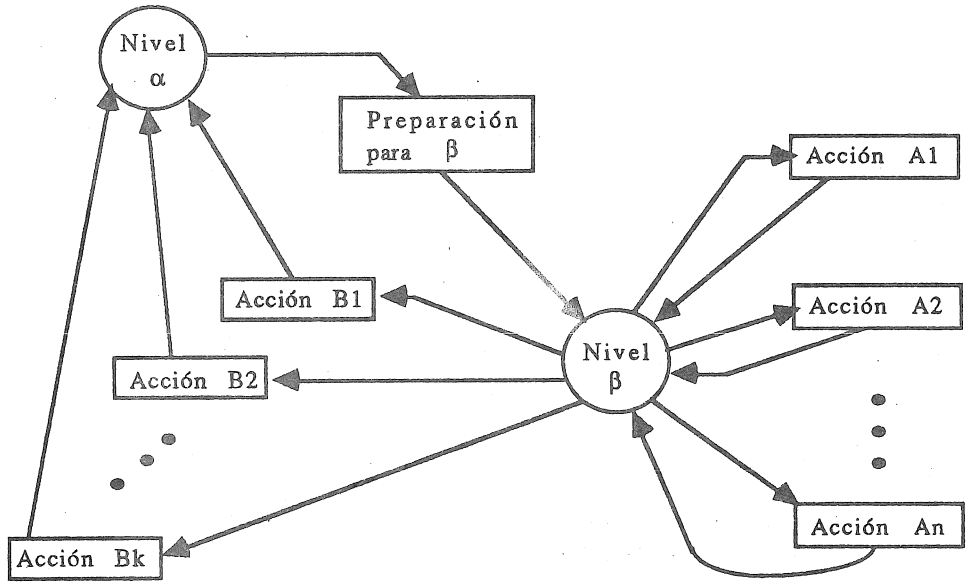


Regla única de Reescritura:

Parte Izquierda:



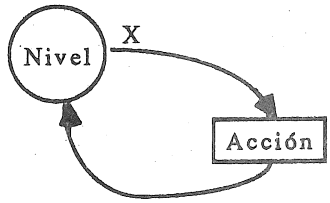
Parte Derecha:



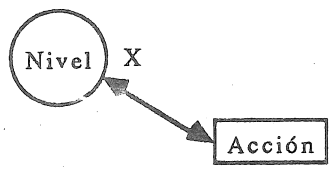
donde:

* k ≥ 1, es decir, existe al menos una transición de regreso al Nivel padre;
* existe una transición A_i ó B_j encargada de procesar el caso de "error", es decir, la selección por parte del usuario de una acción no disponible en ese nivel.

NOTA: En lo sucesivo



se abreviará así



---*---

Como ilustración de las reglas anteriores, a continuación se muestra un Diagrama Estructurado de Transiciones que presenta un subconjunto de WordStar®:

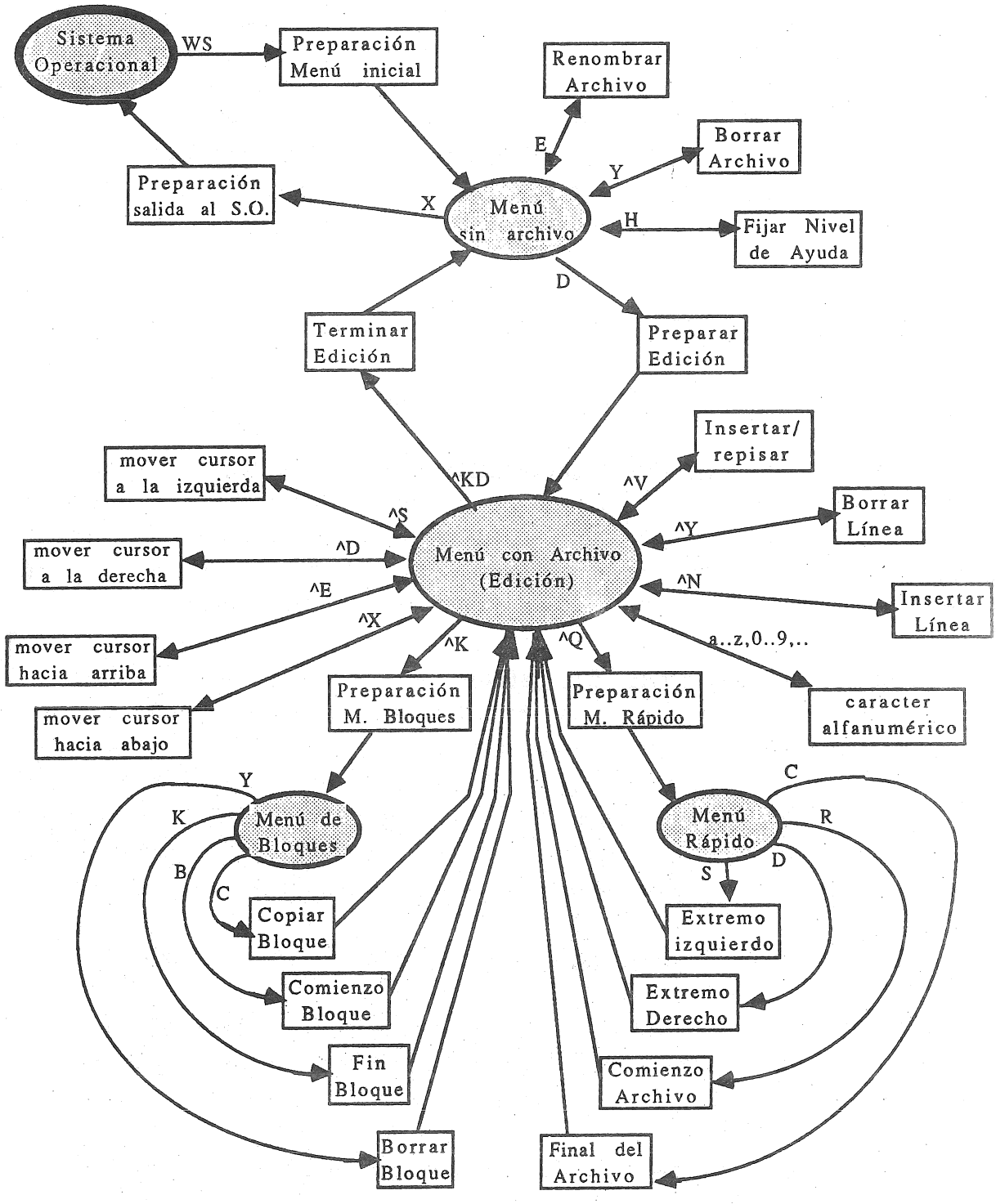


Figura Nº 1: Diagrama de Transiciones de un subconjunto de WordStar®

3. ESPECIFICACION DETALLADA DE SISTEMAS INTERACTIVOS

Los Diagramas de Transiciones permiten describir globalmente un sistema interactivo, es decir, establecer qué servicios y en qué contexto estarán disponibles para el usuario. Sin embargo, para poder obtener un mayor provecho de la labor de diseño, es necesario detallar la forma y apariencia de cada uno de los Niveles, y el efecto de cada uno de los Procesadores de Comando. Esta labor de describir en detalle **qué** hacen las partes de un sistema sin explicitar aún el **cómo** es lo que se conoce con el nombre de "Especificación".

3.1 ESPECIFICACION DE LOS NIVELES

Especificar un Nivel significa establecer claramente:

* El conjunto de Comandos accesibles: Es de resaltar que esto no significa que todos los comandos de un nivel estén accesibles en todo momento. Probablemente para poder activar ciertos comandos sea necesario que el usuario haya llevado el sistema a determinadas condiciones (p. ej.: para poder "borrar bloque" se requiere que haya un "bloque marcado").

* El conjunto de Variables de Estado: son aquellas que indican la situación de los objetos que se manipulan en dicho Nivel en un momento determinado. Se distinguen porque serán frecuentemente consultadas y/o modificadas por varios de los Procesadores de Comandos que se desprenden del Nivel. Ejemplos de variables de estado de un sistema interactivo de edición de texto podrían ser: 'posición del cursor', 'margen izquierda', 'insertando o repisando', 'total de líneas', ..., y particularmente 'texto en edición'.

  Dos observaciones son importantes con respecto a estas Variables de Estado:

  - Un Nivel hereda las Variables de Estado asociadas a sus Niveles ancestro, es decir, sus Procesadores de Comando pueden consultar y/o modificar las Variables de Estado de los Niveles superiores.

  - La manera más cómoda y eficiente de implementar las Variables de Estado es con variables globales (i.e. accesibles desde cualquier parte del programa). Esto evita colocar siempre los mismos parámetros en muchos procedimientos, lo cual recarga enormemente la sintaxis de un programa.

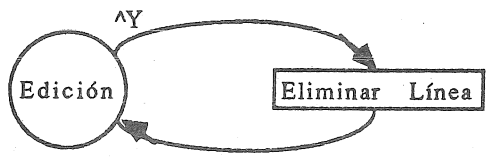
* La Especificación de la Pantalla: Desde un punto de vista meramente computacional, la mayor parte del código de un programa interactivo está dedicado a dibujar, refrescar o actualizar la pantalla, de manera tal que ésta refleje adecuadamente la acción de los comandos que ha dado el usuario. Se requiere entonces especificar con detalle la configuración y disposición de la pantalla en cada uno de los Niveles. Se distinguen cuatro componentes en la especificación de la pantalla asociada a un Nivel:

  - **Parte Fija**: menú, ayudas o mensajes fijos que siempre aparecen en la pantalla;
  - **Parte Variable**: es la imagen de Variables de Estado del Nivel o de Niveles superiores (p.ej.: porción del texto en edición, posición del cursor, posiciones de tabulación, ...);
  - **Parte Entorno**: es la imagen del entorno en el cual se está ejecutando el programa (p.ej.: disco activo, directorio de archivos, tipo de terminal, ...);
  - **Interfase usuario↔ sistema para selección de comandos**: especificación de los mecanismos de que dispone el usuario para seleccionar el comando que desea (p. ej.: selección directa de comandos; preselección, cambios de preselección y confirmación; preselección anticipada del comando más frecuente, etc.).

* Aserciones de Corrección del Nivel: Cada Nivel tiene asociado un conjunto de Variables de Estado (propias y heredadas) y una configuración particular de la pantalla (la cual probablemente incluye la imagen de Variables de Estado). Se pueden entonces establecer Aserciones de Corrección del Nivel, es decir, afirmaciones que caractericen el comportamiento correcto del sistema (p.ej.: "el trozo de texto que aparece en pantalla debe estar alineado según la variable de estado 'margen izquierda'").

3.2 ESPECIFICACION DE LOS PROCESADORES DE COMANDOS

Los Procesadores de Comando deben ser especificados en el contexto de la Transición en la cual se encuentran. Tomando como base de ejemplo la Transición que se muestra a continuación, especificar un Procesador de Comando significa establecer claramente los siguientes aspectos:



* El Nivel de Origen.

* El Nivel de LLegada, eventualmente el mismo de origen.

* La PreCondición: esta condición se construye con la aserción de corrección asociada al Nivel de Origen más un "bautizo" de la situación en la que se encuentran las Variables de Estado que serán afectadas por el Procesador de Comando en cuestión (p.ej. " el sistema está disponible para seguir editando" Y "el texto tiene N líneas" Y "el cursor se encuentra en la línea k del texto" Y "el cursor aparece en la fila f, columna c de la pantalla" ...).

* La PostCondición: en general esta condición tiene la forma de la siguiente conjunción: "aserción de corrección asociada al nivel de destino" Y "se efectuó la acción que había seleccionado el usuario sobre las variables de estado" Y "se reflejó en pantalla dicho cambio en las variables de estado" (p. ej.: "el sistema está disponible para seguir editando" Y "el texto tiene N-1 líneas" Y "el texto ya no tiene la linea k" Y "se suprimió de la pantalla la fila f" Y "el cursor aparece en la fila f, columna 1 de la pantalla" ...).

* El Selector: es el evento que debe causar el usuario para que se active la ejecución del Procesador de Comando en cuestión. Puede ser simplemente digitar un caracter o una cadena, o más generalmente, la ocurrencia de un evento más complejo causado por el usuario (p.ej. "click del ratón en una zona determinada de la pantalla").

4. "ALGORITMO" PARA DISEÑAR SISTEMAS INTERACTIVOS

Los aspectos mencionados en los numerales anteriores sobre Diagramas Estructurados de Transiciones, especificación detallada de Niveles y especificación detallada de Procesadores de Comando, pueden integrarse en un "algoritmo" de diseño de sistemas interactivos.

El punto de arranque del "algoritmo" es el símbolo distinguido, en el cual el sistema interactivo que se planea diseñar aparece como un único Procesador de Comando dependiente del Nivel 0 (el sistema operacional). Los pasos del "algoritmo de diseño" son entonces los siguientes:

Para cada procesador de comandos que no sea de tipo "batch", es decir, cuya ejecución requiera la intervención reiterada y decisiva del usuario, reemplazarlo por un nuevo Nivel (véase la 'Regla única de reescritura', numeral 2), y especificar:

i. **Tipo Abstracto asociado al Nivel**: Como se mencionó anteriormente, los Niveles son materializaciones de Tipos Abstractos de Datos (i.e. objeto + operaciones asociadas). Con frecuencia, con solo explicitar el tipo abstracto asociado al Nivel, se sugieren operaciones apropiadas para dicho tipo abstracto [BOAR 84, cap. 3].

   Por ejemplo, si en algún Nivel tuvieramos el tipo abstracto "*Conjunto de* facturas", sea cual sea el tipo de aplicación que se esté diseñando muy posiblemente se requerirán comandos como '*adicionar* factura', '*retirar* factura', '*seleccionar un subconjunto de* facturas', '*modificar* factura', ...

ii. **Variables de Estado asociadas al Nivel**: Estas variables se reconocen porque frecuentemente deberán ser consultadas y/o modificadas por los diversos Procesadores de Comando que se desprenden del Nivel. El conjunto de Variables de Estado asociadas a un Nivel está compuesto por aquella(s) que constituye(n) el objeto que se manipula en dicho Nivel más las variables que determinan la situación de manipulación.

   Por ejemplo, algunas de las variables de estado asociadas al nivel "Menú con Archivo - (edición)" que aparece en la Figura 1 podrían ser: 'texto en edición', 'fila del cursor', 'columna del cursor', 'primera línea mostrada en pantalla', 'última línea mostrada en pantalla', 'insertando o repisando', 'total de líneas', ... [BORL 86].

iii. **Valor Inicial de las Variables de Estado**: Una vez enumeradas las Variables de Estado asociadas a un Nivel, se debe establecer el valor inicial que éstas deben tomar cada vez que se acceda a dicho Nivel proveniente de un Nivel superior. Continuando con el mismo ejemplo del Nivel "menú con Archivo - (edición)" (Fig. 1), los siguientes podrían ser valores iniciales plausibles para las Variables de Estado mencionadas en el párrafo anterior:

```
'texto en edición':= contenido del archivo seleccionado
'fila del cursor':= 1
'columna del cursor':= 1
'total de líneas':= # de líneas del archivo
'insertando o repisando':= insertando
'primera línea mostrada en pantalla':= 1
'última linea mostrada en pantalla':= min  ( MAX_LINEAS_PANTALLA,
                                             # de líneas del archivo seleccionado ).
```

Es importante anotar que las labores de inicializar de las Variables de Estado y de dibujar el pantallazo inicial asociado a un Nivel son responsabilidad del procedimiento "Preparación para ..." que por construcción siempre aparece antes de cada Nivel (véase 'Regla única de reescritura', Numeral 2).

iv. **Enumeración de los Procesadores de Comando que se desprenden del Nivel:** En este punto deberán enumerarse los Procesadores de Comandos que estarán accesibles a partir del Nivel en cuestión. Muy posiblemente, aquí se requerirá ajustar especificaciones hechas en los anteriores puntos ii y iii.

v. **Especificación de la Pantalla asociada al Nivel:** Tal como se explicó anteriormente (numeral 3.1), en este punto es necesario especificar la Parte Fija, Parte Variable, y Parte Entorno que deberán aparecer en la pantalla, así como las características del interfase usuario↔ sistema para la selección de comandos.

vi. **Especificación rigurosa de los Procesadores de Comando tipo "batch" que se desprenden del Nivel:** Una vez enumerados los Procesadores de Comandos, las Variables de Estado y la disposición de la Pantalla, es posible entrar a especificar los diferentes Procesadores de Comandos en términos de PreCondiciones y PostCondiciones. En el numeral 3.2 ya se explicó cual era la forma general de estas aserciones.

5. COMPILACION DE DIAGRAMAS ESTRUCTURADOS DE TRANSICIONES

Una vez que un Sistema Interactivo ha sido diseñado, especificado y validado, se puede proceder a escribir los programas que van a implementar tal diseño. Es aquí donde se manifiesta una de las grandes ventajas de los Diagramas Estructurados de Transiciones: es posible compilarlos inmediatamente para obtener el "Esqueleto de Control" del sistema diseñado. Por **Esqueleto de Control** se entiende el programa que captura los comandos del usuario, despacha el control al Procesador de Comando o Nivel que corresponda, y posteriormente cuando recupere el control despacha el siguiente comando del usuario.

En los siguientes numerales se presentan varios esquemas canónicos de compilación de niveles para diversos tipos de interfase usuario↔ sistema.

Obviamente existen muchos más esquemas de interfase usuario↔sistema que no son cubiertos con los casos que se presentan a continuación. Sin embargo, se ha visto que prácticamente todos los esquemas responden a los mismos principios.

5.1 *Compilación Canónica de Niveles con retorno único y Selección Directa*

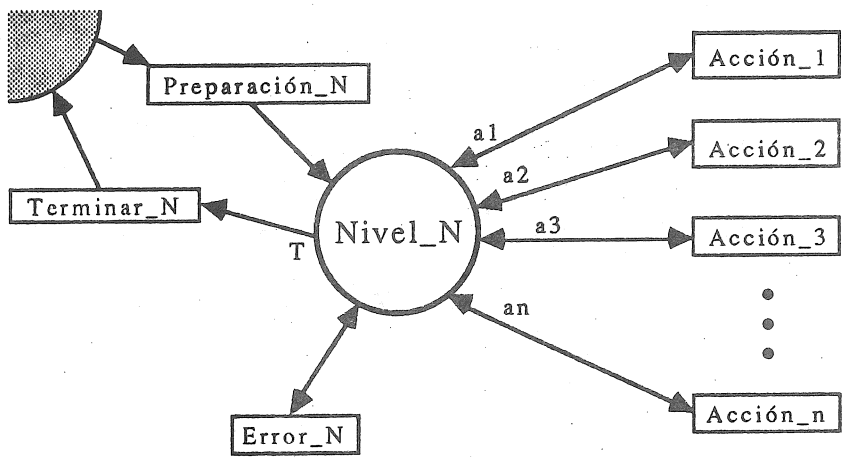


Figura Nº 2: Nivel con retorno único

Supongamos que al Nivel_N del anterior subdiagrama se le asocia un mecanismo muy simple de Selección Directa: cada comando está identificado por una letra, y el usuario selecciona el comando deseado tecleando la letra respectiva. En estas condiciones, el subdiagrama de la Figura Nº 2 se compilaría así:

```
PROCEDURE Nivel_N;                  /* esqueleto de control de un nivel con */
VAR                                 /* retorno único y selección directa    */
   c  :  char;
BEGIN
Preparación_N;
read(c);
while (c <> 'T') do begin
   case c of
      a1 : Acción_1;
      a2 : Acción_2;
      ..   ...

      an : Acción_n;
      ELSE Error_N;
   end;
   read(c);
end;
Salida_N;
END;
```

5.2 *Compilación Canónica de Niveles con retorno múltiple y Selección Directa*

Consideremos ahora el siguiente SubDiagrama:

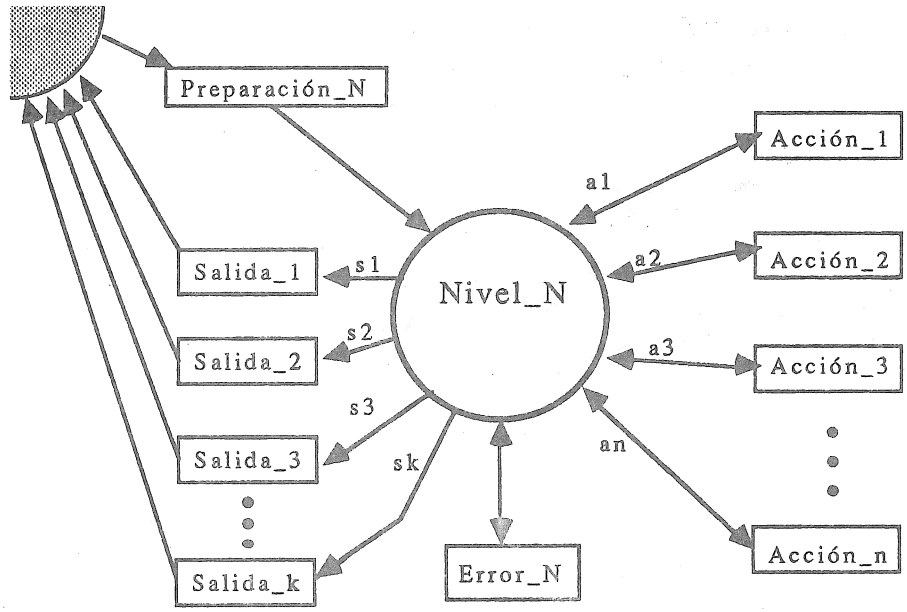


Figura Nº 3: Nivel con retorno múltiple.

Suponiendo de nuevo el mismo esquema de selección directa descrito en el numeral 5.1, el subdiagrama de la Figura Nº 3 se compilaría así:

```
PROCEDURE Nivel_N;                  /* Esqueleto de Control de un Nivel con */
VAR                                 /* retorno múltiple y selección directa */
  c        : char    ;
  fin_N    : boolean;
BEGIN
Preparación_N;
fin_N:= FALSE;
repeat
   read(c);
   case c of
      a1:  Acción_1;
      ..   ...
      an:  Acción_n;
      s1:  begin  Salida_1; fin_N:= TRUE  end;
      ..   ...
      sk:  begin  Salida_k; fin_N:= TRUE  end;
      ELSE Error_N;
   end;
until fin_N;
END;
```

5.3 *Compilación Canónica de Niveles con Retorno Múltiple, Preselección y Selección Directa*

Consideremos de nuevo el subdiagrama de la Figura Nº 3, pero esta vez con un mecanismo un poco más complejo para selección de comandos: Selección Directa más Preselección-Confirmación. En otras palabras, supongamos que en cada interacción el usuario puede:

- cambiar la preselección con teclas de dirección (⬆ ⬇ ⬅ ➡), ó
- activar la ejecución de la acción preseleccionada oprimiendo alguna tecla de confirmación **OK**, ó
- activar la ejecución de cualquier acción oprimiendo su selector directo.

Es deseable que la acción que se encuentre preseleccionada aparezca resaltada en la pantalla. Cuando el usuario cambie de preselección con las teclas de dirección, el resaltado debe desplazarse a la nueva preselección. Adicionalmente, cuando el usuario seleccione una acción oprimiendo su selector directo, el resaltado debe quitarse de donde se encontraba, colocarse sobre la acción que acaba de ser seleccionada directamente, esperar allí algunas décimas de segundo, y posteriormente activar la acción respectiva.

Siguiendo las anteriores descripciones, el subdiagrama de la Figura Nº 3 se compilaría en el procedimiento mostrado en la página siguiente.

---*---

Cabe anotar que los ejemplos de esqueletos de control que se han mostrado son "compilaciones canónicas", es decir, salvo pequeños detalles de sistema operacional, tipo de pantalla o compilador, realmente compilan el esqueleto de control del Nivel en cuestión. Al ciclo principal de estas compilaciones canónicas se le suele llamar *"Despachador de Comandos"*.

Sinembargo, este aspecto de la compilación canónica de Niveles admite un gran número de variaciones. El factor que genera estas variaciones es el tipo de interfase usuario↔ sistema para selección de comandos. De todas maneras, para una buena cantidad de esquemas se ha visto que se puede construir una variante apropiada a partir de alguna de las compilaciones canónicas que se han mostrado anteriormente.

```
PROCEDURE Nivel_N;                    /* Esqueleto de Control de un Nivel con */
VAR                                   /* retorno múltiple,selección directa y */
   c        : char;                   /*      preselección + confirmación     */
   fin_N    : boolean;
   presel   : (a1,a2,...,an,s1,s2,...sk);
BEGIN
Preparación_N;
fin_N:= FALSE;
presel:= a1;
resalte(presel);
repeat
   read(c);
   case c of
      ➡:   SucesorHorizontal(preselec);
      ⬇:   SucesorVertical(preselec);
      ⬅:   PredecesorHorizontal(preselec);
      ⬆:   PredecesorVertical(preselec);
      OK:  case preselec of
              a1:  Acción_1;
              ..   ...
              an:  Acción_n;
              s1:  begin  Salida_1; fin_N:= TRUE  end;
              ..   ...
              sk:  begin  Salida_k; fin_N:= TRUE  end;
           end;
      a1:  begin  arregle(preselec,a1); Acción_1  end;
      ..:  ..;
      an:  begin  arregle(preselec,an); Acción_n  end;
      s1:  begin  arregle(preselec,s1); Salida_1; fin_N:= TRUE  end;
      ..:  ..;
      sk:  begin  arregle(preselec,sk); Salida_k; fin_N:= TRUE  end;
      ELSE Error_N;
   end;
until fin_N;
END;
```

donde:

```
PROCEDURE SucesorHorizontal (VAR  ActualPreselec  :  ...);
BEGIN
normalice(ActualPreselec);
ActualPreselec:= ...{fórmula o tabla para determinar el sucesor };
resalte(ActualPreselec);
END;

{Los procedimientos SucesorVertical, PredecesorHorizontal y
 PredecesorVertical tienen exactamente la misma estructura que el
 anterior. Varían únicamente en la forma de determinación del nuevo valor
 de 'ActualPreselec' (asignación de la 2ª línea)
}
```

```
PROCEDURE arregle (VAR  ActualPreselec  :  ...;
                        NuevaPreselec   :  ... );
BEGIN
normalice(ActualPreselec);
resalte(NuevaPreselec);
ActualPreselec:= NuevaPreselec;
delay( )
END;
```

6. PROTOTIPIFICACION DE SISTEMAS INTERACTIVOS

Uno de los nuevos postulados para el desarrollo de sistemas de software de mediana complejidad es la incorporación directa del usuario en las fases de diseño y especificación. Esto se trata de lograr a través de la elaboración de prototipos rápidos en los que se van explicitando y consensando las necesidades reales de los usuarios. Dichos prototipos deberán evolucionar en sucesivas versiones hasta culminar en el sistema definitivo [BOAR 84][AGRE 86].

Antes de plantear en términos generales la metodología de prototipificación, considérese inicialmente el siguiente Diagrama Estructurado de Transiciones:

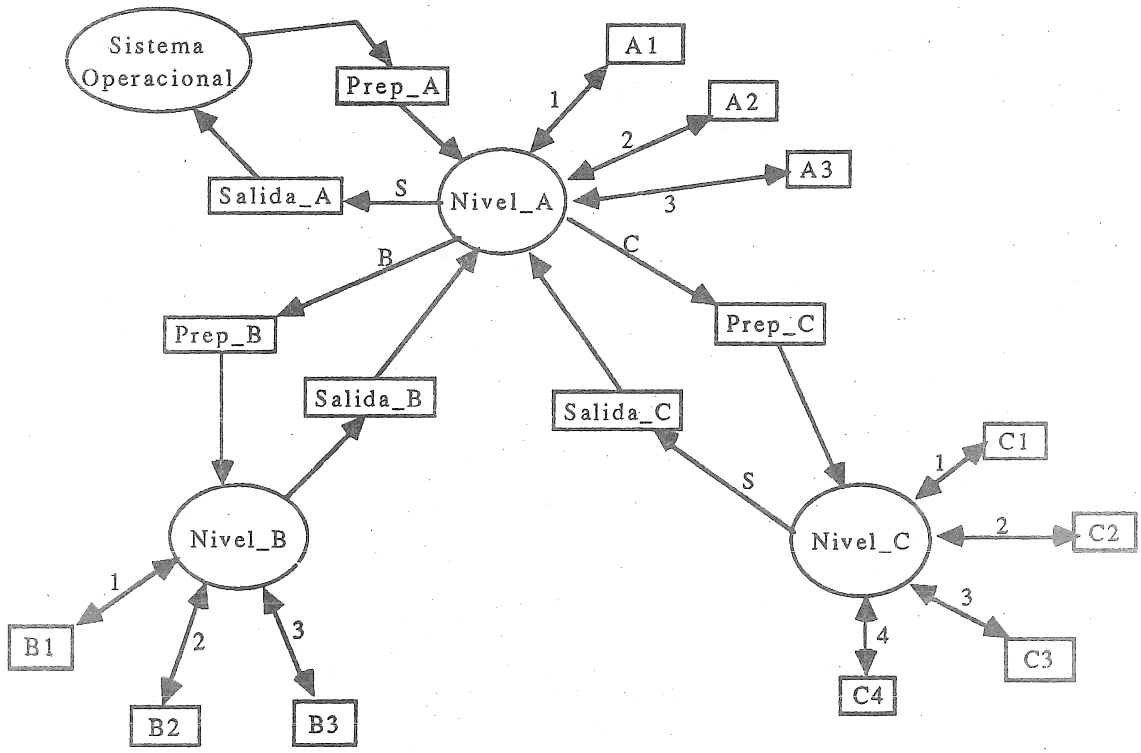


Figura Nº 4: Un Diagrama de Transiciones de profundidad dos

Observando con atención se verá que el siguiente programa resuelve todo el problema de captura y despacho de control asociado al diagrama de la Figura Nº 4 (¡¡ independientemente de cual sea la semántica de los diversos Procesadores de Comando !!).

```
PROGRAM Figura_4 (input, output);    { Esqueleto del Sistema Interactivo }
                                     {      descrito en la Figura Nº 4     }
VAR
   c  :  char;

{-------------------------------------}

PROCEDURE Prep_A;  BEGIN .... END;
PROCEDURE A1;  BEGIN .... END;
PROCEDURE A2;  BEGIN .... END;
PROCEDURE A3;  BEGIN .... END;
PROCEDURE ErrorNivel_A;  BEGIN .... END;
PROCEDURE Salida_A;  BEGIN .... END;

PROCEDURE Prep_B;  BEGIN .... END;
PROCEDURE B1;  BEGIN .... END;
PROCEDURE B2;  BEGIN .... END;
PROCEDURE B3;  BEGIN .... END;
PROCEDURE ErrorNivel_B;  BEGIN .... END;
PROCEDURE Salida_B;  BEGIN .... END;

PROCEDURE Prep_C;  BEGIN .... END;
PROCEDURE C1;  BEGIN .... END;
PROCEDURE C2;  BEGIN .... END;
PROCEDURE C3;  BEGIN .... END;
PROCEDURE C4;  BEGIN .... END;
PROCEDURE ErrorNivel_C;  BEGIN .... END;
PROCEDURE Salida_C;  BEGIN .... END;

{-------------------------------------}

PROCEDURE Nivel_B;
BEGIN
Prep_B;
read(c);
while c <> 'S' do begin
   case c of
      '1': B1;
      '2': B2;
      '3': B3;
      else ErrorNivel_B
   end;
   read(c)
end;
Salida_B
END;

{-------------------------------------}
```

```
{--------------------------------------}

PROCEDURE Nivel_C;
BEGIN
Prep_C;
read(c);
while c <> 'S' do begin
   case c of
      '1': C1;
      '2': C2;
      '3': C3;
      '4': C4;
      else ErrorNivel_C
   end;
   read(c)
end;
Salida_C
END;

{**************************************}
{********  PROGRAMA PRINCIPAL *********}
{**************************************}
BEGIN
Prep_A;
read(c);
while c <> 'S' do begin
   case c of
      '1': A1;
      '2': A2;
      '3': A3;
      'B': Nivel_B;
      'C': Nivel_C;
      else ErrorNivel_A
   end;
   read(c)
end;
Salida_A
END.
```

En términos generales, el desarrollo de un prototipo consta de tres etapas: primero escribir el **Esqueleto del Sistema**, luego definir y maquillar la interfase usuario↔sistema, y posteriormente ir implementando los diversos servicios.

Se toma entonces como punto de partida un Diagrama Estructurado de Transiciones suficientemente especificado (véase numeral 3), y se procede a compilar el Nivel superior (i.e. aquel único Nivel que desciende directamente del Sistema Operacional). El código obtenido se constituye en el programa principal. Posteriormente se procede a compilar cada uno de los otros Niveles del Diagrama. Se irá obteniendo así una serie de procedimientos despachadores de control que son invocados o bien por el programa principal o bien por otros procedimientos despachadores de control.

Obviamente, el encapsulamiento del programa principal más los procedimientos despachadores de control obtenidos al compilar todos los Niveles no constituye aún un programa fuente completo. Hace falta resolver todas las referencias a los procedimientos Procesadores de Comando que figuran en los códigos obtenidos al compilar los Niveles. Para ésto, se debe proceder a crear un procedimiento, inicialmente vacío, por cada Procesador de Comando que aparezca en el Diagrama de Transiciones. El programa obtenido al encapsular el programa principal, los procedimientos que compilan los Niveles, y los procedimientos vacios correspondientes a los Procesadores de Comandos constituye el Esqueleto del **Sistema Interactivo**.

El programa Esqueleto obtenido es compilable y ejecutable, y permite que el control visite cualquier nivel o procesador de comandos. Sin embargo, aún no permite que el usuario valide la información desplegada en los diferentes niveles, ni perciba y evalue la apariencia de la interfase. Para esto, como mínimo sería necesario rellenar los procesadores de comandos que conectan dos Niveles diferentes con el código necesario para desplegar el pantallazo del Nivel de llegada (en el ejemplo de la Figura Nº 4, estos serían los procedimientos 'Prep_A', 'Prep_B', 'Prep_C', 'Salida_A', 'Salida_B' y 'Salida_C'). De esta manera, se obtendría un primer prototipo 'cascarón', que permite visitar cualquier Nivel o Procesador de Comandos, y facilita que el usuario se forme una idea global del tipo y forma de la información que se desplegará en pantalla en cada Nivel.

En esta primera fase de prototipificación es importante postergar al máximo la producción de código atado a un tipo específico de interfase o a una distribución particular en la pantalla de la información presentada en un Nivel. De hecho, son justamente los aspectos de interfase y presentación de la información los que suscitan el mayor número de sugerencias o críticas de los usuarios durante ltodo el desarrollo del sistema.

Para obviar este problema hay un método muy sencillo que nos ha dado excelentes resultados: con un editor de texto normal se crean una a una configuraciones plausibles de los diversos pantallazos, los cuales son guardados en archivos separados. Posteriormente se coloca en cada uno de los procedimientos que conectan Niveles una invocación del estilo:

```
MuestrePantallazo ( nombre del archivo que contiene el pantallazo asociado al Nivel de llegada ) ;
```

donde 'MuestrePantallazo' simplemente copia en pantalla el contenido del archivo especificado. Adicionalmente, se pueden establecer algunos caracteres de control para indicar cambios de color o de atributos de video, los cuales serían interpretados por 'MuestrePantallazo' al momento de estar haciendo el eco a la

pantalla. De esta manera, bastaría con editar y modificar el archivo para reflejar los cambios de forma que sugiera el usuario. Posteriormente, partir de este primer prototipo "cascarón" pueden irse desarrollando en forma evolutiva versiones cada vez más sofisticadas del prototipo, hasta converger al sistema definitivo [BOAR 84][RIVE 87].

CONCLUSIONES

En este trabajo se ha presentado un lenguaje para describir globalmente Sistemas Interactivos. Posteriormente se han descrito cada uno de los pasos que se deben recorrer para diseñar y especificar dichos sistemas. En seguida se mostró como se pueden procesar tales diseños en forma cuasi-automática para obtener el Esqueleto del Sistema Interactivo. Finalmente, se vió como se puede construir muy rápidamente un prototipo "cascarón" que permita al usuario tener un primer contacto "real" con el sistema en desarrollo, siendo evidente que posteriores refinamientos de dicho prototipo pueden incorporarse ordenadamente en el Esqueleto del Sistema Interactivo.

La piedra angular de la metodología son los Diagramas Estructurados de Transiciones. Estos Diagramas, definidos rigurosamente por una gramática formal, establecen claramente el espacio de posibilidades en el que mueve el diseñador. Es importante anotar que otros trabajos están explorando extensiones "non sanctas" a dichos diagramas, algunas de las cuales son de gran utilidad para el tratamiento y recuperación de errores [LOPE 87b]. Está aún pendiente la especificación y el esquema de compilación cuasi-automática de situaciones que frecuentemente ocurren en la práctica, como por ejemplo la anulación o interrupción por parte del usuario de la ejecución normal de los procesadores de comando.

Si bien todos los ejemplos de código se hicieron en Pascal, cabe anotar que las compilaciones canónicas admiten codificaciones muy elegantes en el lenguaje C. De todas maneras, la similitud de los diversos esquemas canónicos de compilación de Niveles sugiere una extensión hacia manejadores generalizados de Niveles. De hecho, en 'C' pueden escribirse en forma condensada, elegante y eficiente manejadores generalizados que pueden procesar diversos Niveles vía una adecuada parametrización [KALY 85].

Actualmente el grupo "ISAC: Ingeniería de Software Apoyada por Computador" está trabajando en la construcción de un ambiente de soporte a la metodología aquí descrita. Tales soportes incluyen un editor/compilador de Diagramas de Transiciones, un editor/compilador de pantallazos [RIVE 87], así como diversas librerías de manejo de interfases.

BIBLIOGRAFIA

[AGRE 86] Agresti, W. W.
"Tutorial: New Paradigms for Software Developpment"
IEEE Computer Society Press 1987
Código IEEE: EH0245-1; Código ISBN: 0-8186-0707-6

[BARN 86] Barnard, J.; Metz, R.; Price, A.
"A Recommended Practice for Describing Software Designs: IEEE Standards Project 1016"
IEEE Transactions on Software Engineering, Vol. SE-12, Nº 12
Febrero 1986

[BERG 81] Bergland, G.
"A Guided Tour for Program Design Methodologies"
IEEE Computer, Vol. 14, Nº 10, Octubre 1981

[BOAR 84] Boar, B. H.
"Application Prototyping: a requirement definition strategy for the 80's"
John Wiley & Sons, 1984

[BOHO 86] Bohórquez, J. A.
"Cálculo de Programas: una Metodología para el Diseño de Programas de Computador"
- UniAndes-CIFI, Memo de Investigación Nº 6, 1986
- XII Conferencia Latinoamericana de Informática CLEI, Montevideo, Noviembre 1986

[BORL 86] Borland International Inc.
"Turbo Editor Toolbox - Owner's Handbook" ©1985

[CARD 86] Cardoso, R.
"Diseño e Implementación de Tipos Abstractos de Datos: una Metodología basada en Funciones Recursivas"
- UniAndes-CIFI, Memo de Investigación Nº 9, 1986
- XII Conferencia Latinoamericana de Informática CLEI Montevideo, Noviembre 1986

[CARD 87] Cardoso, R.
"Prácticas en Tipos Abstractos de Datos"
- UniAndes-CIFI, Memo de Investigación Nº 17, 1987
- XII Conferencia Latinoamericana de Informática CLEI
Montevideo, Noviembre 1986

[GEHA 86] Gehani, N. (Ed.)
"Software Specification Techniques"
Addisson Wesley, 1986

[GRIE 81] Gries, D.
"The Science of Programming"
Springer Verlag, 1981

[KALY 85] Kalyan, D.
"Modular Programming in C: an approach and an example"
Sigplan Notices, Vol. 20, Nº 3, pp. 9 - 15, Marzo 1985

[LOPE 87a] López, R.; Toro, V. M.
"turbo-KAREL (versión 1.5): Libro del Proyecto"
Informe final del Contrato FC-009
Fonade-Secretaría de Informática de la Presidencia de la República-
Universidad de los Andes. Mayo de 1987.

[LOPE 87b] López, R.
"Saltos no Locales: una extensión a Turbo Pascal"
XIII Conferencia Latinoamericana de Informática
Bogotá, noviembre de 1987

[MYER 78] Myers, G.
"Composite Structured Design"
Van Nostrand, 1978

[PRES 87] Pressman, R. S.
"Software Engineering: a Practitioner's Approach" (2ª edición)
McGraw Hill, 1987

[RIVE 87] Rivera, G.; Reyes, I. H.
"Cartelera Dinámica: diseño e implementación"
XIII Conferencia Latinoamericana de Informática
Bogotá, noviembre de 1987

[YOUR 76] Yourdon, E.; Constantine, L.
"Structured Design: Fundamentals of a Discipline of Computer Programs and System Design"
Prentice-Hall (Yourdon Press), 1976

[YOUR 86] Yourdon, E.
"What ever happened to Structured Analysis ?"
Datamation, junio 1986, págs. 133-138

[WASS 81] Wasserman, A. I.
"User Software Engineering and the Design of Interactive Systems"
5th International Conference Software Engineering (IEEE)
San Diego CA, marzo 1981.

---*---