METODOLOGIA DE CONSTRUCCION DE PROGRAMAS A PARTIR DE ESPECIFICACIONES FORMALES

M. RUKOZ *

RESUMEN

El objetivo de este trabajo es la elaboración de una metodología de construcción de programas que permita de una forma sistemática elaborar programas confiables.

La metodología se basa por una parte en la utilización de operadores genéricos, que permiten hacer abstracciones operacionales y por otra parte en la representación algoritmica de Tipos de Datos Abstractos, que permitan hacer abstracciones en la estructura de datos.

ABSTRACT

This paper intented to develop a program construction methodology in order to elaborate reliable software in a systematic manner. Methodology is based on utilization of generic operator and algorithmical representation of abstract data types, that make possible the action and object abstractions respectively.

* Licenciada en Computación (Vla, 1981)
Universidad Central de Venezuela, U.C.V.,
Facultad de Ciencias, Escuela de Computación,
Apartado 47002, Caracas 1041-A, Venezuela.
Tlf. (02) 6627233

INTRODUCCION

Dentro del proceso de diseño e implantación de sistemas de computación, uno de los principales problemas es la reducción de complejidad que la persona debe afrontar en un momento dado para la solución del mismo. Una vía para resolver esto es a través de la abstracción, esto es, desarrollar modelos que revelen las propiedades relevantes e ignoren los detalles irrelevantes del sistema.

La resolución algoritmica de problemas implica el diseño de un algoritmo, basado en cierta metodología que permita una solución (si existe) y en la decisión de la organización de la información relevante al mismo. Estos dos factores están íntimamente relacionados, de manera tal que las decisiones acerca de la estructuración de los datos requiere conocer del algoritmo, así como el algoritmo depende de tal estructuración.

Así podemos distinguir dos puntos en la solución algoritmica de un problema donde se puede hacer uso de la abstracción: el algoritmo y la estructura de datos. Surgen así dos mecanismos de abstracción: abstracción operacional y abstracción de datos, según (Bert 82a) 'The action abstraction' y "The object abstraction' respectivamente.

El objetivo de este trabajo es la elaboración de una metodología de construcción de programas que permita de una forma sistemática elaborar programas confiables.

La metodología se basa por una parte en la utilización de operadores genéricos, que permiten hacer abstracciones operacionales y por otra la representación algoritmica de tipos de Datos Abstractos, que permiten hacer abstracción en la estructura de datos.

En dicha metodología se especifica un problema funcionalmente, lo cual permite un análisis completo del mismo a nivel de especificaciones, para luego transformarlo en un programa algoritmico utilizando esquemas de programación establecidos.

En la primera sección, se plantea un lenguaje que permita especificar operadores genéricos y Tipos de Datos Abstractos, y como se puede especificar funcionalmente un problema utilizando dicho lenguaje. En la segunda sección se dan esquemas de programas algoritmicos para cada uno de los constructores del lenguaje dado y por último en la tercera sección se define la metodología.

1. PRESENTACION DEL LENGUAJE DE ESPECIFICACION

Con el fin de obtener especificaciones matemáticamente correctas y relativamente fáciles de elaborar, utilizaremos el lenguaje de especificación y programación LPG (Lenguaje de Programación Genérica) este es presentado en (Bert 83) y actualmente está implementado en el Laboratorio IMAG de la Universidad de Grenoble, Francia.

Este lenguaje está basado en las especificaciones algebráicas de Tipos de Datos Abstractos (Gut 78a), (Gut 78b) y permite definir de una manera abstracta los tipos de datos y el conjunto de operadores.

En LPG tanto los tipos de datos como los operadores pueden ser genéricos, es decir, parametrizados por otros tipos llamados tipos formales. Un tipo formal está caracterizado por

un conjunto mínimo de operadores que deberá poseer el tipo efectivo. Este conjunto mínimo de operadores puede ser identificado con el nombre de propiedad y se definen semánticamente a través de axiomas ecuacionales.

Además de los operadores genéricos LPG presenta otros constructores básicos que permiten especificar problemas, estas son: la condición, la composición y la sustitución.

A continuación se muestran como se especifican algebráicamente los tipos de datos abstractos, la definición de los constructores básicos y como se definen los operadores genéricos en LPG.

1.1 Definición de Tipo de Datos Abstractos en LPG

Tipo nombre-del-tipo

opns
  { identificación sintáctica de los operadores}

Vars
  { declaración de variables }

equs
  { identificación semántica de los operaciones}

1.2 Constructores básicos de LPG

Condición: Si b ENTONCES A SINO B FSI

Composición: F; G

Sustitución: SEA w = C EN B(w) fSEA

donde: b, A, B, F, C son constructores de LPG; w es una variable; B(w) es un constructor que contiene w.

1.3 Definición de Operadores Genéricos en LPG

Enriq : nomb-del-enriq exige nombre-de-propiedad ($t_1,..t_n$ opns $0_1..0_n$)

opns
  {identificación sintáctica de los operadores }

vars
  {declaración de variables }

equs
  {lista de ecuaciones que identifican semánticamente a los operadores }

fin

donde: $t_1,..t_n$ son los tipos efectivos sobre los cuales se exige la propiedad

$0_1,..0_n$ son los operadores efectivos

Prop nombre-de-propiedad sobre $t_1, \ldots, t_n$ (tipos formales)

opns

{identificación sintáctica de los operadores formales}

vars

{declaración de variables}

equs

{lista de ecuaciones que identifican semánticamente a los operadores}

fin

Para expresar funcionalmente un problema se le han asociado a los constructores básicos y a los operadores genéricos de LPG una forma funcional. En la tabla I se muestran las formas funcionales de los diferentes operadores a considerar.

Las formas funcionales permiten hacer manipulaciones en las especificaciones, en la tabla II se presentan diferentes reglas que permiten pasar de una forma funcional a otra, en estas reglas se ha hecho abstracción de las variables que utiliza una determinada forma funcional.

FORMAS FUNCIONALES DE LOS CONSTRUCTORES DE LPG

Condición Si b ENTONCES A sino B FSI (x) = =

Si b(x) ENTONCES A(x) sino B(x) FSI

Composición F o G (x) = = F(G(x))

Sustitución SEA w = A EN B(w) FSEA (x) = = B(A(x), x)

Operadores Genéricos

- phi $\{f_1, f_2\}$ $(x,y) = = (f_1(x), f_2(y))$
- mu $\{f_1, f_2\}$ $(x,y) = = (f_1(f_2(x), y))$
- psi $\{f_1, f_2\}$ $(x,y) = = (f_1(x), f_2(x))$
- iter $\{f_1, f_2\}$ $(x, \langle y_1, y_2, \ldots y_n \rangle) = =$ SEA $w = f_2(x, y_1)$
  EN $\pi_1(w) \rightarrow$ iter $\{f_1, f_2\}$ $(\pi_2(w), \langle y_2, \ldots, y_n \rangle)$
  FSEA

  iter $\{f_1, f_2\}$ $(x, \langle : t_1 \rangle) = = f_1(x)$
- arra $\{f_1, f_2, f_3\}$ $(x_1, \langle y_1, y_2, \ldots y_n \rangle)$ SEA $w = f_2(x_1, y_1)$
  EN $f_3(w,$ arra $\{f_1, f_2, f_3\}$ $(\pi_2(w), \langle y_2, \ldots y_n \rangle)$
  FSEA

  arra $\{f_1, f_2, f_3\}$ $(x_1, \langle : t_1 \rangle) = = f_1(x)$

TABLA I (Continúa)

TABLA I (Continuación)

- alpha $\{f\}(\langle x_1,\cdots,x_n\rangle) = = \langle f(x_1),f(x_2),\cdots,f(x_n)\rangle$
- $\partial'\{f,x\} = = f(x)$
- $\$'\{f,x\}\ (y) = = f(x,y)$
- $C'\{x\}\ (y) = = x$
- $S\{i\}(\langle x_1,x_2,\cdots x_n\rangle) = = x_i$ si $1 \leq i \leq n$, sino indefinido
- $hom\{c,f\}(\langle x_1,x_2,\cdots x_n\rangle) = = f(x_1,hom'\{c,f\}(\langle x_2,\cdots,x_n\rangle))$
  $hom\{c,f\}(\langle :t\rangle) = = c$

NOTA: $\langle :t\rangle$ es la secuencia vacía de elementos de tipo t
! 1 y ! 2 son funciones definidas de la manera siguiente:

$!1: (t_1,t_2) \to t_2$; $!2: (t_1,t_2) \to t_2$; a: t_1, b: t_2

!1(a,b) = = a y !2(a,b) = = b

REGLAS DE TRANSFORMACION

1. id o g = = g o id = = g
2. (f o g) o h = = fo(g o h)
3. Si b ENTONCES y SINO q FSI oh = = Si b o h ENTONCES y o h SINO q o h FSI
4. h o Si b ENTONCES r SINO q FSI = = Si b ENTONCES h o r SINO h o q FSI
5. Si b ENTONCES Si b ENTONCES p SINO q FSI SINO r FSI = =
   Si b ENTONCES p SINO r FSI
6. Si b ENTONCES r SINO q FSI = = Si NO(b) ENTONCES q SINO r FSI
7. Si C'{VERDAD} ENTONCES r SINO q FSI = = r
8. Si C'{FALSO} ENTONCES r SINO q FSI = = q
9. Si b ENTONCES C {VERDAD} SINO p FSI = = OR o psi {b,p}
10. ! 1 o Psi $\{f_1,f_2\} = = f_1$
11. C'{h} o g = = C'{h}
12. ! 2 o psi'$\{f_1,f_2\} = = f_2$
13. ! 1 o phi'$\{f_1,f_2\} = = f_1$ o !1
14. !2 o phi'$\{f_1,f_2\} = = f_2$ o !2
15. phi'$\{f_1,f_2\}$ = = psi'$\{f_1$ o !1, f_2 o !2$\}$
16. phi'$\{f_1,f_2\}$ o psi'$\{f_3,f_4\}$ = = psi'$\{f_1$ o f_3, f_2 o $f_4\}$
17. phi'$\{f_1,f_2\}$ o phi'$\{f_3,f_4\} = =$ phi'$\{f_1$ o f_3, f_2 o $f_4\}$
18. mu'$\{f_1,f_2\}$ o psi'$\{f_3,f_4\} = = f_1$ o psi'$\{f_2$ o f_3, $f_4\}$

TABLA II

A cada constructor funcional de LPG (constructores básicos y operadores genéricos) se le ha definido un esquema algoritmico que permita el paso casi mecânico de las especifica - ciones al algoritmo final del problema.

A continuación se presentan los diferentes esquemas de programación:

```
Esq   (Si  b  ENTONCES  A  SINO B  FSI)
ent   x: tl  Sal  R: t2
inst  Si Sal (esq (b(x))) entonces  R: = sal(esq(A(x)))
                          Sino  R: = sal (esq(B(x)))  fsi
Esq (f(g(x))
ent  x:tl  sal R:t2  Var  y:t
inst  y:= sal(esq(g(x))); R:= sal(esq(f(y)))
Esq  (SEA  w = A EN  B(w)  FSEA )
ent  x:t  sal  R:t2
inst  w:= sal (esq(A(x)) ; R:= sal(esq(B(w,x))
Esq (phi { f1,f2 } )
ent   X:tl, y:t2   sal  R1:t3 , R2:t4
inst  R1: = sal (esq(f1(x))) ; R2:= sal (esq(f2(y)))
Esq (mu { f1,f2 } )
ent  X:tl  y:t2  sal  R:t3  var  X2:t3
inst  X2:= sal (esq(f2(x))); R:= sal(esq(f1(X2,y)))
Esq  (psi { f1,f2 } )
ent  X:tl  sal  R1:t2, R2:t3
inst  R1: = sal(esq(f1(X))) ; R2 : = sal(esq(f2(X)))
Esq  (iter { f1,f2 } )
ent  P1:t3 ; S1: seq < tl > sal  S2 : seq < t2 >
var  X:tl, SF2: seq < t2 >,PF2 : t3
inst  PF2:= P1;
     mientras  no (nul(S1)) hacer
        X:=PRIMERO(S1) ; S1:=RESTO (S1)
        (SF2,PF2) : = sal (esq(f2(PF2,X))) ;
         S2 : = S2 + SF2
     Fmientras
     S2: = S2 + sal (esq(f1(PF2)))
```

```
Esq (arra { f1,f2,f3} )
  ent  P1 : t3   S1 : seq < t1 > ; sal   S2 :  seq < t2 >   P2:t3
  var  SF2: seq < t2 > ,  PF2 : t3   X : t1
  inst   S2 : = nul ; P2 : = P1 ; PF2 : = P1 ;
         mientras   no(nul(S1))  hacer
            X : =  PRIMERO (S1) ; S1 : = RESTO (S1)
            (SF2,PF2) : = sal (esq(f2(PF2,X))
            (S2,P2) : =   sal (esq(f3(S2,P2,SF2,PF2)))
         fmientras
         (S2, P2 ) : = sal (esq(f1 (P2)))

Esq (Alpha { f } )
ent  S1 : seq < t1 > sal  S2: seq < t2  >, var   X:t1
inst  S2: = nul
      mientras  no(nul(S1))  hacer
         X : = PRIMERO (S1) ; S1 : = RESTO (S1) ; S2 : = S2 + sal(esq(f (X))))
      fmientras

Esq ( ∂ { f } )
   ent  X : t1 , sal  y : t2
   inst      y : = sal (esq(f(X)))

Esq  hom  { c,f } )
   ent  S : seq  <t >  sal   d:dom   var  X:t ,   d1:dom
   inst      Si  nul(S) entonces   d : = c
             Sino
               X : = PRIMERO (S) ; S : = RESTO (S) ;
               d1 : = sal(esq(hom(S))) ; d : = sal (esq(f(X,d1)))
             fsi
```

3. METODOLOGIA PROPUESTA

La metodología consta de los pasos siguientes:

a. Análisis del problema y elaboración de especificaciones informales.

b. Estudio de las especificaciones informales para determinar:
   - Los Tipos de Datos Abstractos
   - Los operadores importantes

c. Especificar algebráicamente los Tipos de Datos Abstractos, utilizando la notación dada en la Sección I.

d. Especificar funcionalmente los operadores importantes utilizando los constructores funcionales dados en la Sección I.

e. Escribir los operadores en forma funcional, utilizando las formas funcionales asociadas a los constructores y optimizarlos utilizando las reglas de transformación dadas en la tabla II.

f. Elaborar una interfase algoritmica abstracta con los Tipos de Datos Abstractos, que permita la conexión entre los Tipos de Datos Abstractos y el programa algoritmico que los utiliza.

g. Transformar las especificaciones de los operadores en programas algoritmicos, a través de los esquemas de programación de los constructores funcionales dados en la sección II y de la interfase algoritmica abastracta elaborado en el paso 6 de la metodología.

4. CONCLUSIONES

Hemos presentado una metodología de construcción de programas que permite en una forma sistemática elaborar programas confiables, centrando todo su interés en escribir todo el problema una primera vez, a través de un lenguaje sin ambiguedades como lo es el de especificación funcional, lo cual permite visualizar la arquitectura del programa completo. De esta forma se da la idea de todo un proyecto, sus dificultades, sus divisiones lógicas, etc.

La escritura completa de un proyecto en una forma precisa antes de su programación permite hacer modificaciones, estructuraciones, descomposiciones, etc., a nivel de especificaciones.

Los puntos f y g de la metodología permiten de una manera fácil pasar de las especificaciones a una representación algoritmica.

5. BIBLIOGRAFIA

(BACK 78): J. BACKUS
Can programming be liberated from the Von Neuman Style? A functional Style and its Algebra of Programs. CACM Vol. 21, N8 1978.

(BERT 82a) D. BERT
Software Components Construction, in Tool and Notions for Program Construction. Ed. D. Neal. Cambridge University Press. Pag. 347-376.

BERT 82b) D. BERT
Generic Programming: A tool for designing universal operators. Aplication to program Algebra. RR 336 Lab. IMAG Grenoble 1982.

BERT 83) D. BERT
Manual de reférence de LPG. RR 458 Lab. IMAG Grenoble 1983.

(GUT 78a) J.V. GUTTAG; E. HOROWITZ; D.R. MUSSER
Abstract Datas Types and Software validation. CACM Vol. 21 n 12 1978.

(GUT 78b) J.V. GUTTAG; J.J. HORNING
The Algebraics Specification of Abstract Data Types. Acta Informática 10 1978

(GOG 78) J.A. GOGUEN; J.W. THATCHER; E.G. WAGNER
An Initial Algebra Aproach to the Specification, Correctness
and Implementation of Abstract Data Types. Current Trends in
Programming Methodology. Vol. 4, Data Structuring Prentice Hall 1978