# Translatoare catre limbaje grafice

Ec. Robert ENYEDI
Alpha Tech 2000

*The software translation process is mainly oriented towards the conversion of legacy software systems to modern systems. Visual source code analysis tools are a category of software translators with an application area that differs from that of traditional software translators. These tools have certain architectural characteristics and present various levels of analysis detail.*

***Keywords:*** *software translation, compiler technologies, graphical languages.*

## Notiuni generale de translatare software

Evolutia limbajelor de programare a cunoscut o mare varietate de reprezentanti, grupate pe generatii. Noile generatii de calculatoare împreuna cu sistemele de operare au dus la aparitia unor noi limbaje de programare si la marginalizarea altora. De asemenea, repartitia specialistilor în diferitele limbaje de programare si arhitecturi s-a modificat în decursul timpului.

Din punctul de vedere al structurarii, limbajele de programare pâna în prezent au evoluat în trei mari etape: limbaje nestructurate, limbaje structurate si limbaje orientate obiect.

Evolutia a avut loc în sensul cresterii gradului de organizare a codului sursa, ceea ce este important atât din punctul de vedere al dezvoltarii software cât si din cel al mentenantei software. Limbajele de nivel scazut au reprezentanti doar în etapa limbajelor nestructurate. Restul etapelor au cunoscut limbaje de programare de nivel înalt, compilate sau interpretate.

Limbajele de nivel înalt au ajuns predominante si doar în rare cazuri este nevoie de serviciile oferite de un limbaj de nivel scazut. Motivul este ca limbajele de nivel înalt, prin însasi definitia lor, au o expresivitate net superioara fata de cele de nivel scazut în conditiile în care resursele de calcul disponibile sunt incomparabil mai bogate decât în trecut.

Fiecare perioada a informaticii a dus la raspândirea anumitor tipuri de limbaje de programare. În decursul timpului, sisteme informatice dezvoltate au ajuns sa devina învechite. Acest context a dus la aparitia translatoarelor software si a translatoarelor de limbaje.

Un translator de limbaje construieste, pe baza unui program sursa A scris în limbajul L, codul sursa A' în limbajul L' astfel încât programul A este echivalent cu programul A' (figura 1).

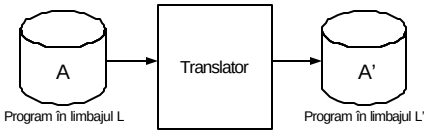



**Fig. 1.** Procesul de translatare automata a programelor

## Limbajele grafice ca tinta a procesului de translatare

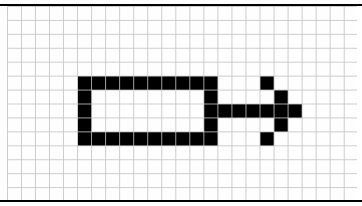
Prezentul articol abordeaza problematica translatarii software dintr-un alt unghi decât cel general întâlnit. Este vorba despre subcategoria utilitarelor destinate procesului de dezvoltare software pentru analiza vizuala a codului sursa.

Activitatea de dezvoltare de programe, ca baza a industriei software, este un proces de creatie realizat în limitele impuse de specificatiile proiectului. Codul sursa ce sta la baza oricarui sistem informatic cunoaste un proces de acumulare pe masura ce proiectul parcurge etape de dezvoltare si se apropie de produsul final. Limbajele de programare de nivel înalt si în special limbajele de programare orientate obiect ofera un nivel superior de organizare a logicii aplicatiei, dar este necesar un continuu proces de organizare si întretinere a bazei de cod sursa.

Pentru realizarea acestei activitati au fost dezvoltate variate utilitare ce permit o vizua-

lizare grafica a structurii proiectului, la variate nivele de detaliere. Aceste utilitare în esenta sunt translatoare de limbaje si difera de acceptiunea uzuala prin faptul ca limbajul tinta este unul grafic. Codul sursa translatat este transformat din algoritmi într-un set de primitive grafice în plan: puncte, linii si curbe. Folosind aceste elemente, translatorul defineste obiecte grafice de nivel ridicat cum ar fi: dreptunghiuri, sageti, text sau imagini.
În functie de formatul de afisare, în unele cazuri este vorba de un limbaj formal complet iar în altele limbajul este neformal ce rezulta din logica codului sursa a modulului de desenare. Din punct de vedere semantic în fiecare dintre cazuri avem de a face cu un limbaj grafic (Tabelul 1).

**Tabelul 1.** Exemple de limbaje grafice

| Limbaj grafic formal | Limbaj grafic neformal | Rezultat |
|---|---|---|
| Line (5,5),(14,5)<br>Line (14,5),(14,9)<br>Line (14,9),(5,9)<br>Line (5,9),(5,5)<br>Arrow (15,7),(20,7) | g.drawLine(5,5,14,5);<br>g.drawLine(14,5,14,9);<br>g.drawLine(14,9,5,9);<br>g.drawLine(5,9,5,5);<br>drawArrow(g,15,7,20,7); |  |

**LogicalSW – translator catre un limbaj grafic**

Orice program informatic este o descriere formala a unui algoritm, sub forma unei secvente de propozitii validate de gramatica limbajului folosit. Practica programarii a promovat textul ca forma concreta pentru scrierea programelor. Astfel, orice program este o secventa de caractere ordonate în functie de logica aplicatiei si conform regulilor gramaticii limbajului de programare utilizat (de aici notiunea de program sau cod sursa).
În mod uzual, ca modalitate de exprimare a algoritmilor într-o forma vizuala se folosesc schemele logice. Schema logica este o modalitate de a reprezenta un algoritm într-o forma mai sintetica decât textul.
LogicalSW este un utilitar de translatare catre un limbaj grafic destinat generarii automate a schemelor logice din programe sursa. Limbajul sursa este limbajul C iar limbajul tinta este un limbaj grafic neformal. Aplicatia lucreaza pe fisiere de cod sursa si genereaza o schema logica per fisier sursa (figura 3).
Arhitectura LogicalSW se bazeaza pe arhitectura unui compilator si include: un parser top-down recursive descent, constructorul de arbore de sintaxa si motorul de renderizare a schemei logice.
Parserul top-down, pe baza programului sursa, comanda constructorului de arbore de sintaxa sa compuna arborele de sintaxa (figura 2). Algoritmul pe care se bazeaza parserul este recursive descent. Acesta parcurge gramatica limbajului C începând de la simbolul de start (unitatea logica cea mai cuprinzatoare, adica programul) si, coborând catre simbolurile terminale (tokens), realizeaza o identificare a regulilor gramaticale continute în codul sursa. În cazul în care codul sursa nu se conformeaza gramaticii limbajului C, este generat un mesaj de eroare cu referinta catre pozitia exacta din text care a generat eroarea iar constructia schemei logice este oprita.
Deoarece schema logica are reguli de corectitudine mult mai putin stricte decât programul sursa, pentru a realiza construirea schemei logice nu este necesar un proces de analiza semantica a textului sursa (în general analiza semantica cuprinde analiza de tip si analiza domeniului de valabilitate a identificatorilor).
Motorul de renderizare a schemei logice reparcurge arborele de sintaxa si deseneaza schema logica folosind un limbaj grafic neformal. Obiectele grafice utilizate sunt blocuri, sageti si elemente de legatura. Blocurile sunt de cinci tipuri (tabelul 2):

- blocurile de început si sfârsit de program;
- blocuri de comanda simpla desenate pentru o instructiune simpla;
- blocuri de apel de functie pentru expresii si instructiuni ce contin în corpul lor cel putin un apel de functie;

- blocuri de intrare-iesire pentru operatiile intrare si iesire;
- blocuri decizionale pentru instructiuni decizionale si de ciclu conditionat.


**Fig. 2.** Reprezentare intermediara sub forma de arbore de sintaxa

**Tabelul 2.** Tipurile de blocuri

| Tip bloc | Exemplu |
|---|---|
| Bloc de început si sfârsit de program | START / STOP |
| Bloc de comanda simpla | a= a* 2 |
| Bloc de apel de functie | vdf= fd(x0) |
| Bloc de intrare-iesire | scanf("%f", &eps) |
| Bloc decizional | !vb |

Un exemplu de translatare catre limbaj grafic realizat de LogicalSW este cel din figura 3. Programul prezentat este implementarea în C a metodei bisectiei pentru rezolvarea ecuatiilor liniare. Exemplul de arbore de sintaxa prezentat în figura 2 a fost generat pentru acest cod sursa.

```
/*Linear equation solver
using the bisection method.*/
#include <stdio.h>
#include <math.h>
void main()
{
  float x1;
  float x2;
  float eps;
  float xmed;
  scanf("%f",&x1);
  scanf("%f",&x2);
  scanf("%f",&eps);
  do
  {
    xmed=(x1+x2)/2;
    if(f(x1)*f(x2)<=0)
    {
       x2=xmed;
    }else
    {
      x1=xmed;
    }
  }while(abs(x1-x2)>eps);
  printf("solution: %f",xmed);
}
```

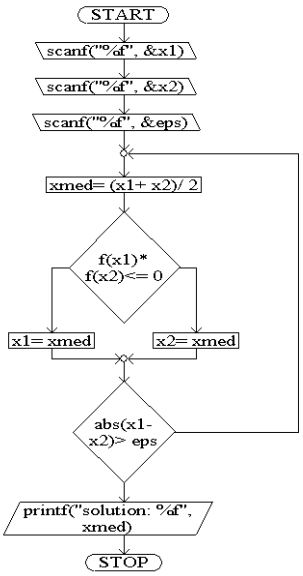



**Fig. 3.** Exemplu de convertire în limbaj grafic realizat de LogicalSW

Se observa cum, pentru a construi schema, au fost eliminate elemente din codul sursa ce nu sunt importante din punctul de vedere al algoritmului. Acesta este cazul comentariilor, instructiunilor de preprocesare si al declaratiilor de variabile. Schema logica rezultata este reprezentarea fidela a algoritmului ce a stat la baza construirii codului sursa. Pe baza schemei logice, întelegerea codului sursa este mult simplificata iar utilitatea schemei logice este mai mare pe masura ce numarul liniilor de cod sursa creste.

## Concluzii

Problematica translatarii software a fost ridicata din necesitatea de a continua exploatarea

sistemelor informatice mostenite fara a apela la reconstruirea manuala a sistemului. Prin proiectarea si dezvoltarea de instrumente de translatare automata devine posibila migrarea sistemelor informatice existente pe sisteme hardware si software moderne, cu costuri relativ reduse si într-un timp relativ scurt.

Practica dezvoltarii software a determinat o directie separata de dezvoltare a translatoarelor software în directia limbajelor grafice. Astfel au aparut utilitarele de ana-liza vizuala a codului sursa extrem de utile în procesul de dezvoltare software. LogicalSW este un astfel de utilitar ce a fost conceput pentru usurarea activitatii de dezvoltare a programelor folosind limbajul C.

**Bibliografie**

[Much97] Steven S. Muchnick – Advanced Compiler Design and Implementation – Academic Press, 1997

[Petz98] Charles Petzold – Programare în Windows 98 – Editura Teora, Bucuresti, 1998

[Smeu98] Ion Smeureanu, Ion Ivan, Marian Dârdala – Structuri si obiecte în C++ – Editura CISON, Bucuresti, 1998

[Watt93] David A. Watt – Programming Language Processors: Compilers and Interpreters – Prentice Hall International, 1993