GRADUL DE INTERFERENȚĂ AL METRICILOR HALSTEAD ŞI McCABE

Ion Ivan
Felix Simion
Viorel Nica

Academia de Studii Economice, Bucureşti

Rezumat: Se prezintă metricile Halstead şi Mc Cabe. Pentru un eşantion reprezentativ format din 50 de programe, se calculează indicatorii ai celor două metrici. Se calculează un indicator normat. Se definesc patru intervale neegale, care definesc grade de complexitate. Pentru lotul considerat, se defineşte apartenenţa indicatorului normat de complexitate la unul din subintervale. Este definit şi calculat gradul de interferenţă care permite identificarea situaţiilor în care se utilizează una sau alta dintre metrici, fără a afecta calitatea interpretării rezultatelor.

Cuvinte cheie: metrici software, complexitate programe, grad interferenţă.

1. Introducere

Pentru complexitatea software-ului, sunt numeroase metrici dintre care metrica Halstead şi metrica McCabe, prin simplitate şi obiectivitate, au cea mai largă întrebuinţare [3].

În lucrare, se prezintă cele două metrici, programe pentru implementarea lor şi modalităţi de evidenţiere a interferenţei dintre acestea.

2. Metrica Halstead

În [1] a fost prezentată, pentru prima dată, această metrică sintetizând rezultate fundamentale care pot fi obţinute prin analiza textului sursă al unui set de produse-program. Halstead a considerat, pe baza unor cercetări făcute în timp, ca unele caracteristici ale produselor-program pot fi cuantificate.

În continuare, vom folosi următoarele notaţii:

- n_1 - numărul de tipuri fundamentale de date care apar distinct în program;
- n_2 - numărul de tipuri derivate de date care apar distinct în program;
- n_3 - numărul de instrucţiuni distincte utilizate de programator;
- n_4 - numărul de operanzi distincţi care apar în program;
- n_5 - numărul de operatori distincţi pentru referire care apar în program;
- n_6 - numărul de funcţii distincte apelate.

Pe baza notaţiilor de mai sus pot fi calculaţi următorii indicatori:

- complexitatea produsului-program (notată cu **C**);
- efortul de programare (notat cu **E**);
- volumul produsului-program (notat cu **V**);
- nivelul program (notat cu **L**).

Acesti indicatori au urmatoarele formule de calcul :

$$C = \sum_{i=1}^{6} n_i \log_2 n_i \ ;$$

Considerând $\eta_1 = n_1 + n_2$ şi $\eta_2 = n_3 + n_4 + n_5 + n_6$ putem scrie că avem

$$C = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$$

Efortul de programare se calculeaza dupa urmatoarea formula :

$$\mathbf{E} = \frac{V}{L}$$

unde avem

$V = N\log 2 \sum_{i=1}^{6} n_i$, $N = \sum_{i=1}^{6} n_i *$ n_i* având aceeaşi semnificatie ca şi n_i , dar contorizează totalurile, nu numai pe cele distincte.

Aceste formule au creat premisa scrierii unui produs-program care automatizează procesul de evaluare a unor programe test prin calculul cu o eroare cât mai mică a metricei Halstead.

În continuare, vom prezenta, pe scurt, produsul-program.

Acesta trebuie privit ca un analizator lexical pe textul sursă al produselor software scrise în C/C++. Modul de lucru al analizatorului este sintetizat în etapele următoare:

etapa nr.1: este caracterizată de modul de introducere al lotului de programe pentru care se doreşte calcularea indicatorilor. Programul analizează dacă toate fişierele text cu extensia *.CPP care formează baza de programe de lucru se află pe disc. În caz contrar, utilizatorul este atenţionat printr-un mesaj de eroare;

etapa nr.2: este specifică analizatorului lexical şi se împarte în următoarele:

- se elimină toate comentariile din textul sursă al programului analizat;
- în cazul în care programatorul a folosit biblioteci individuale de produse-program (de exemplu: #include "progr1.cpp" sau #include "calcul.h") se trece automat la analiza acestora,

deoarece indirect afectează rezultatul final al indicatorilor ;

- se gestionează toate instrucțiunile folosite în cadrul programului analizat ;
- se gestionează funcțiile utilizate făcându-se distincție între cele definite de utilizator și cele ale mediului de programare C/C++. S-a ținut cont de faptul că acest mediu de programare este puternic orientat pe funcții ;
- se analizează fiecare expresie în parte, evidențiidu-se operanzii, operatorii, conversiile implicite. .

```
{
double p;
int i;
for(i=p=1;i<=n;++i) p*=x;
return p;
}

double puterer(double x ,int n)
{
return (n>0)?x*puterer(x,n-1):1;
}
```

Caracteristicile programului sunt prezentate în următorul tabel:

Operator	Contor1	Operand	Contor2	Funcție	Contor3
( )	18	x	9	puterei( )	3
;	17	n	9	puterer( )	3
=	3	p	4	printf( )	4
*	1	I	4	scanf( )	2
?	1	*Total*	26	getch( )	2
:	1			*Total*	14
<=	1				
Total	42				

etapa nr.3: se obține un tabel cu rezultate finale din care utilizatorul analizatorului lexical poate determina o ierarhie a produselor-program analizate, folosind diverse criterii de scalare (complexitate, lungime program, efort de programare etc.).

Ca un rezultat experimental, se poate considera următorul program care realizeaza ridicarea la putere a unui număr, iterativ și recursiv.

```
#include<stdio.h>
#include<conio.h>
double puterei(double,int);
double puterer(double,int);
void main()
{
double x;
int n;
printf("\n Dati numarul de ridicat la putere : ");
scanf("%lf",&x);
printf("\n Dati exponentul :"); scanf("%d",&n);
printf("\n Rezultatul iterativ este : %lf",
puterei(x,n));
getch();
printf("\n Rezultatul recursiv este:
%lf",puterer(x,n));
getch();
}

double puterei( double x, int n)
```

Deci, am obținut următoarele valori pentru variabile:

- $n_1 = 2$; $n_1^* = 15$
- $n_2 = 0$; $n_2^* = 0$
- $n_3 = 2$; $n_3^* = 2$
- $n_4 = 4$; $n_4^* = 26$
- $n_5 = 7$; $n_5^* = 42$
- $n_6 = 5$; $n_6^* = 14$

Pe baza acestora, am obținut următoarele valori pentru indicatori:

C = 39.267;

V = 436.607;

E = 16601.026.

3. Complexitatea ciclomatica

O dată cu acceptarea pe scara largă a programarii structurate ca metoda eficienta de dezvoltare a programelor, a crescut interesul pentru realizarea de metrici software care sa masoare complexitatea structurala. O masura clasica pentru controlul fluxului este complexitatea ciclomatica, dezvoltata de catre cercetatorul american Thomas McCabe [2], avand la baza elemente de teoria grafurilor.

Fiecarui program i se poate asocia un graf orientat, în care nodurile reprezintă instrucțiunile programului, iar arcele corespund ordinii de execuție a instrucțiunilor. De exemplu, programului ce rezolvă problema de numărare a elementelor pozitive, negative şi nule ale unui masiv unidimensional, prezentat mai jos, i se asociază un astfel de graf ca în Figura 1.

```
I0      void main()
        {

        int n=10,np,nn,nz,i,x[10]={-
        1,0,5,3,-4,8,0,6,-6,2};

I1          i=np=nn=nz=0;
I2          do
            {
I3              if(x[I]>0)
I4              np++;
                else
I5              if(x[i]<0)
I6                nn++;
                else
I7                nz++
I8,I9           ;
I10         i++;
I11         }while(i<n);
I12         printf("\n%2d elemente
            pozitive\t%2d elemente
            negative\t%2d elemente\
                    nule",np,nn,nz);
I13     }
```

Observații:

Fiecarei instrucțiuni i s-a ataşat un simbol care se regăseşte ca etichetă de nod în Figura 2. Instrucțiunile I8 şi I9 corespund instrucțiunii vide.

Complexitatea ciclomatică se determină conform relației:

$$C_c = E - n + 2$$

unde: E - numărul arcelor grafului
n - numărul nodurilor grafului

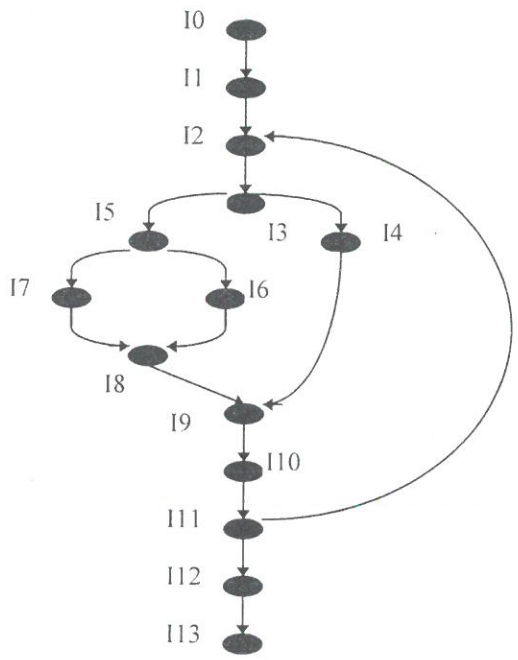

Figura. 1. Graful asociat programului

Pentru graful prezentat în Figura1 n = 14 noduri, iar E = 16 arce, deci complexitatea ciclomatică este 16-14+2 = 4.

Pentru construirea grafului program, s-a realizat aplicația GRAFPROG care analizează texte sursă C/C++ realizând următoarele:

- elimină şirurile de caractere şi comentariile din programul analizat;
- parcurge fişierul sursă, identificând instrucțiunile;
- stochează fiecare instrucțiune într-un nod al unui graf alocat dinamic;
- identifică ordinea de execuție a instrucțiunilor şi o memorează prin intermediul arcelor grafului;
- parcurge graful astfel construit, calculând diverşi indicatori printre care şi complexitatea ciclomatică.

4. Rezultate experimentale

Pentru un eşantion de 50 de programe scrise în C/C++, s-au calculat cu ajutorul programelor METRHALS şi GRAFPROG indicatorii complexității C_H şi C_C, prezentați anterior. În tabelul din **ANEXA Nr. 1,** sunt prezentate rezultatele obținute, pentru fiecare program indicându-se şi dimensiunea sa fizica (în bytes).

Se calculează valorile normate ale indicatorilor dupa următoarele relații:

$$I'_H = \frac{\sum_{i=1}^{6} n_i \log_2 n_i}{\left(\sum_{i=1}^{6} n_i\right) \log_2 \left(\sum_{i=1}^{6} n_i\right)}$$

$$I_C = \frac{E - n + 2}{n^2 - 2n + 2}$$

Grad de interferență

Se consideră intervalele:

I1 = [0,0.5)

I2 = [0.5,0.78)

I3 = [0.78,0.89)

I4 = [0.89,1]

asociate unor niveluri de complexitate a programelor. Se construiesc matricele:

$$M_A = (a_{ij})_{\substack{i=1,N \\ j=1,M}} \quad \text{unde} \quad a_{ij} = \begin{cases} 0, \text{daca } I_{H_i} \notin I_j \\ 1, \text{daca } I_{H_i} \in I_j \end{cases}$$

$$M_B = (b_{ij})_{\substack{i=1,N \\ j=1,N}} \quad \text{unde} \quad b_{ij} = \begin{cases} 0, \text{daca } I_{C_i} \notin I_j \\ 1, \text{daca } I_{C_i} \in I_j \end{cases}$$

$$M_C = (c_{ij})_{\substack{i=1,N \\ j=1,M}} \quad \text{unde} \quad c_{ij} = a_{ij} + b_{ij}$$

Gradul de interferență între cei doi indicatori ai complexității se calculează pentru fiecare interval considerat în parte, cât și pentru total, astfel:

$$K_{I_j} = \frac{\text{card}(D_j)}{\text{card}(U_j)}, j = 1, M \text{ unde:}$$

$$D_j = \{c_{ij} \in M_C \mid c_{ij} = 2, i = 1, N\}, j = 1, M$$

$$U_j = \{c_{ij} \in M_C \mid c_{ij} \neq 0, i = 1, N\}, j = 1, M$$

$$K = \frac{\text{card}(D)}{N}, \text{ unde:}$$

$$D = \{c_{ij} \in M_C \mid c_{ij} = 2, i = 1, N; j = 1, M\}$$

S-au folosit notațiile:

K_{I_j} - gradul parțial de interferență dintre indicatori corespunzător intervalului I_j

K - gradul total de interferență dintre indicatori

N - numărul de intervale

M - numărul de programe din baza de test

Datele experimentale, rezultate pe baza calculelor de mai sus, sunt evidențiate în **ANEXA Nr. 2**.

5. Concluzii

Se observă că cele două metrici oferă posibilitatea de a include un program în aceeași clasă de complexitate cu probabilitatea de 50%. Probabilitatile ca un program să aparțină aceleiași clase de complexitate în sens Halstead și, respectiv, McCabe sunt următoarele:

- pentru clasa de programe *foarte simple*: 35.29%
- pentru clasa de programe *simple*: 38.71%
- pentru clasa de programe *complexe*: 16.67%
- pentru clasa de programe *foarte complexe*: 33.33%

În cazul împărțirii în trei intervale de complexitate, reprezentate pe o scara ierarhica sub forma :

- foarte simple pentru intervalul I1 = [0;0.4);
- simple pentru intervalul I2 = [0.4;0.7);
- complexe pentru intervalul I3 = [0.7;1.0].

probabilitățile sunt următoarele: 25%, 32% și, respectiv, 57.14%.

Dacă se realizează o împărțire în două clase de complexitate, și anume:

- *simplu* pentru intervalul I1 = [0;0.6);
- *complex* pentru intervalul I2 = [0.6;1.0].

avem probabilitățile următoare: 43.48% și, respectiv, 60%.

Bibliografie

1. **HALSTEAD, M. H.**: Elements of Software Science, Elsevier, North-Holland, 1977.
2. **McCabe, T.J.**: A complexity measure. În: IEEE Transaction on Software Engineering SE-2(4), 1976, pp. 308-320.
3. **IVAN, I., MĂCEȘANU, M., ARHIRE, R.**: Clase de complexitate pentru produse-program, Bucuresti, 1985.
4. **IVAN, I., POPESCU, M.**: Metrici software. În: BYTE, vol. 2, nr. 5, 1996, pp. 73-82.

INDICATORI AI COMPLEXITĂȚII

ANEXA 1

Nr. Crt.	Ch	Cc	Ih	Ic	L(Bytes)
1	13583.21	76	0.6704	0.9833	10293
2	704.85	7	0.5926	0.5825	1070
3	594.21	9	0.7272	0.7431	759
4	926.13	4	0.7395	0.8179	1140
5	4086.41	14	0.7864	0.9766	3322
6	929.86	6	0.8975	0.9062	1196
7	8121.58	30	0.9012	0.9378	4240
8	86.44	2	0.2333	0.4768	154
9	807.14	4	0.5667	0.9066	734
10	1357.47	6	0.9296	0.9409	1589
11	297.29	5	0.2074	0.0000	263
12	261.07	4	0.6864	0.3445	286
13	1028.62	4	0.7617	0.7658	979
14	9406.87	55	0.7309	0.9633	7065
15	568.43	5	0.8407	0.7881	559
16	359.41	5	0.0000	0.5104	746
17	6931.07	14	0.7963	0.8774	3430
18	396.22	8	0.0654	0.0905	422
19	2161.77	13	0.6407	0.8687	1703
20	1833.41	16	0.7111	0.8177	2246
21	572.15	6	0.4333	0.5704	712
22	3973.56	20	0.7210	0.9461	3584
23	6887.59	1	0.4630	0.9566	346
24	17607	132	0.6111	0.9704	1198
25	242.21	3	0.4333	0.3314	322
26	393.45	3	0.6160	0.7119	410
27	179.52	5	0.5728	0.5625	377
28	487.3	10	0.5593	0.6261	602
29	1173.65	9	0.5852	0.7724	1399
30	1377.12	10	0.6346	0.8486	1550
31	1492.48	9	0.9358	0.8435	1238
32	1828.46	31	0.6778	0.8305	2532
33	134.61	2	0.4605	0.3617	229
34	554.66	10	0.7111	0.7137	664
35	1002.85	16	0.7568	0.7430	978
36	640.16	5	0.5593	0.6067	504
37	20752.28	3	0.9568	1.0000	13264
38	286.95	4	0.3333	0.2333	268
39	371.16	4	0.5975	0.6530	365
40	1361.25	16	0.2765	0.6325	1166
41	568.17	7	0.3519	0.7413	683
42	749.87	10	0.3012	0.7307	854
43	568.8	9	0.2111	0.7078	732
44	206.13	2	0.6185	0.5632	271
45	13990.95	54	1.0000	0.9839	9433
46	1563.92	15	0.4494	0.8072	1790
47	10264.31	97	0.5086	0.9745	8927
48	4122.23	17	0.7432	0.9618	2926
49	1096.41	13	0.3938	0.7924	1230
50	652.67	8	0.2272	0.8201	998

INDICATORI AI COMPLEXITĂȚII

ANEXA 2

Nr. crt.	INTERVALE HAL STEAD				INTERVALE McCABE				MATRICE SUMA			
	I1	I2	I3	I4	I1	I2	I3	I4	I1	I2	I3	I4
1	0	1	0	0	0	0	0	1	0	1	0	1
2	0	1	0	0	0	1	0	0	0	2	0	0
3	0	1	0	0	0	1	0	0	0	2	0	0
4	0	1	0	0	0	0	1	0	0	1	1	0
5	0	0	1	0	0	0	0	1	0	0	1	1
6	0	0	0	1	0	0	0	1	0	0	0	2
7	0	0	0	1	0	0	0	1	0	0	0	2
8	1	0	0	0	1	0	0	0	2	0	0	0
9	0	1	0	0	0	0	0	1	0	1	0	1
10	0	0	0	1	0	0	0	1	0	0	0	2
11	1	0	0	0	1	0	0	0	2	0	0	0
12	0	1	0	0	1	0	0	0	1	1	0	0
13	0	1	0	0	0	1	0	0	0	2	0	0
14	0	1	0	0	0	0	0	1	0	1	0	1
15	0	0	1	0	0	0	1	0	0	0	2	0
16	1	0	0	0	0	1	0	0	1	1	0	0
17	0	0	1	0	0	0	1	0	0	0	2	0
18	1	0	0	0	1	0	0	0	2	0	0	0
19	0	1	0	0	0	0	1	0	0	1	1	0
20	0	1	0	0	0	0	1	0	0	1	1	0
21	1	0	0	0	0	1	0	0	1	1	0	0
22	0	1	0	0	0	0	0	1	0	1	0	1
23	1	0	0	0	0	0	0	1	1	0	0	1
24	0	1	0	0	0	0	0	1	0	1	0	1
25	1	0	0	0	1	0	0	0	2	0	0	0
26	0	1	0	0	0	1	0	0	0	2	0	0
27	0	1	0	0	0	1	0	0	0	2	0	0
28	0	1	0	0	0	1	0	0	0	2	0	0
29	0	1	0	0	0	1	0	0	0	2	0	0
30	0	1	0	0	0	0	1	0	0	1	1	0
31	0	0	0	1	0	0	1	0	0	0	1	1
32	0	1	0	0	0	0	1	0	0	1	1	0
33	1	0	0	0	1	0	0	0	2	0	0	0
34	0	1	0	0	0	1	0	0	0	2	0	0
35	0	1	0	0	0	1	0	0	0	2	0	0
36	0	1	0	0	0	1	0	0	0	2	0	0
37	0	0	0	1	0	0	0	1	0	0	0	2
38	1	0	0	0	1	0	0	0	2	0	0	0
39	0	1	0	0	0	1	0	0	0	2	0	0
40	1	0	0	0	0	1	0	0	1	1	0	0
41	1	0	0	0	0	1	0	0	1	1	0	0
42	1	0	0	0	0	1	0	0	1	1	0	0
43	1	0	0	0	0	1	0	0	1	1	0	0
44	0	1	0	0	0	1	0	0	0	2	0	0
45	0	0	0	1	0	0	0	1	0	0	0	2
46	1	0	0	0	0	0	1	0	1	0	1	0
47	0	1	0	0	0	0	0	1	0	1	0	1
48	0	1	0	0	0	0	0	1	0	1	0	1
49	1	0	0	0	0	0	1	0	1	0	1	0
50	1	0	0	0	0	0	1	0	1	0	1	0