Empirical Software Optimization

Prof.dr. Ion IVAN, prep. Cătălin BOJA
Catedra de Informatică Economică, A.S.E. Bucureşti
ionivan@ase.ro, catalin.boja@ie.ase.ro

The paper describes various methods to assess software optimization from the viewpoint of a set of improved software versions of a same product. Based on building a hierarchy the process analyzes and measures the software versions. It computes an aggregate indicator that allows direct comparison of the different software products. There are also described methods to determine selected software characteristics importance weight in the overall quality level of a software product. These are used in combination with the data set obtained by measuring product characteristics levels. The software versions implement various combinations of methods that improve specified software characteristics. Entire optimization process is based on measuring the versions characteristics real levels and to select the one that solves the problem in the most efficient way.
Keywords*: software, optimization, empirical, assessment, hierarchy.*

Optimizarea software

Optimizarea software este un domeniu foarte lung al ingineriei software şi o etapă importantă în dezvoltarea produselor software.

Se consideră un număr dat de programe care rezolvă aceeaşi problemă. Dintre ele *Programul optim* este acela care dă cea mai bună valoarea pentru un indicator numit criteriu de performanţă.

A optimiza un program înseamnă a îmbunătăţi performanţele acestuia, cu pretenţia că s-a obţinut pentru moment o valoare mai bună a criteriului de performanţă. Soluţia nu este unică întru-cât de la o optimizare la alta se ameliorează performanţa. În plus optimizarea software are caracter local, referindu-se la un program care se modifică sau la rezultatul comparării unui număr foarte mic de programe între ele, programe care reprezintă versiuni de rezolvare a unei probleme.

Optimizarea este precedată de realizarea aplicaţiei informatice şi aducerea ei la o formă funcţională lipsită de orice fel de erori. Procesul de optimizare se aplică unei aplicaţii software funcţionale care rezolvă corect şi complet problema pentru care a fost realizată. Optimizarea nu înseamnă corectarea erorilor. Obiectivul acestei etape este constituit de îmbunătăţirea caracteristicilor aplicaţiei prin aducerea lor la un nivel optim.

În ansamblu activitatea de optimizare include sau este strict legată de procesele de:

- evaluare a aplicaţiei; imaginea obiectivă şi completă a aplicaţiei software de optimizat este obţinută în urma măsurării directe, prin rularea cu seturi de date diferite şi în condiţii de lucru reale, a nivelurilor caracteristicilor de calitate analizate;
- testare a versiunilor aplicaţiei; odată definite criteriile de îmbunătăţit şi soluţiile de atingere a acestor obiective se realizează variante diferite de module, secvenţe, algoritmi, tipuri agregate de date care să fie implementate;
- ierarhizare a versiunilor îmbunătăţite; aplicaţia software reprezintă o construcţie complexă cu toate că prin intermediul abordării modulare are asociată o imagine mult simplificată;
- alegerea versiunii optime; în urma analizei comparate a metodelor de optimizare se stabileşte calea de optimizare a aplicaţiei astfel încât aceasta să fie îmbunătăţită pe ansamblu.

Se stabileşte poziţia etapei de optimizare şi caracterul ciclic în ciclul de viaţă al produsului după ce s-a obţinut software funcţional conform figurii 1.

În etapa de realizare a aplicaţiei, specialiştii ce asigură managementul calităţii software stabilesc pe baza analizei caracteristicile sof-

tware considerate esențiale pentru viitorul produs software. Se obțin niveluri estimate şi se definesc obiectivele de realizat. Întregul proces de producţie urmăreşte atingerea obiectivului final, realizarea aplicației, precum şi asigurarea nivelului de calitate propus.

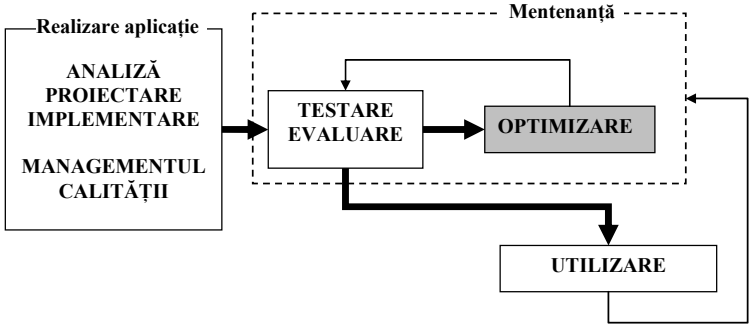

Fig. 1. Ciclul de viață a aplicației

Mai întâi s-a dezvoltat optimizarea pe cod sursă care include:
- eliminarea subexpresiilor comune;
- eliminarea invarianților;
- eliminarea codului mort;
- reducerea expresiilor indiciale;
- regruparea structurilor de control.

Ideea de a optimiza se concretizează prin modificări în programe care reduce volumul de prelucrări. Volumul de prelucrări, dat ca număr de iterații se calculează pentru secvența:

```
S = 0;
for(int i=0; i<n; i++ )
        S+= x[i];
```

Volumul de prelucrări V, este V = 1 + n(1+1+1) = 3n+1 iterații.

Îmbunătățirea secvențelor conduce la micşorarea volumului de instrucțiuni. De exemplu, pentru generarea matricei unitate:

```
for(int i=0;i<n;i++)
        for(int j=0;j<m;j++)a[i][j] = 0;
for(i=0;i<n;i++) a[i][i] = 1;
```

care are un volum $V_1 = n + 2n^2 + 2n = 2n^2 + 3n$. Secvența îmbunătățită este

```
for(int i=0;i<n;i++)
        for(int j=0;j<m;j++)a[i][j] = (i = =j)
```

cu un volum $V_2 = n + n^2$.

Comparând secvențele S_1 şi S_2 rezultă că secvența S_2 este mai performantă. Sunt situații în care se urmăreşte scăderea gradului de complexitate.

Pentru eliminarea subexpresiilor:

$e = \dfrac{a*a + b*b + c*c}{a*a + b*b + c*c + d*d - 1}$, complexitatea este:

$C_1 = 18\log_2 18 + 19\log_2 19$. Dacă înlocuim elementele subexpresiilor cu x = a*a + b*b + c*c + d*d şi e = x / (x-1), atunci complexitatea întregii secvențe este $C_1 = 13 \log_2 13 + 12 \log_2 12$.

Evaluarea expresiei $C_1 > C_2$ conduce la concluzia că eliminarea subexpresiilor conduce la scăderea complexității.

Se dă un lot de programe $P_1, P_2, \ldots, P_n$ care rezolvă aceeaşi problemă şi se consideră criteriile de optim $O_1, O_2, \ldots, O_m$. Se efectuează măsurători şi se construieşte tabelul în care a_{ij} reprezintă nivelul pentru caracteristica O_j măsurat la programul P_i.

Dacă se consideră un singur criteriu de maxim se ia programul care are nivel maxim pentru criteriu şi acesta se consideră programul optim PrO, exclusive în raport cu mulțimea de programe analizate. Dacă se ia în considerare un criteriu de minim, programul cu valoarea cea mai mică este considerat de asemenea optim în raport cu colectivitatea analizată.

Problema alegerii programului optim în raport cu mai multe criterii este o problemă de selecție multicriterială.

Criterii de optim

Există numeroase abordări a performanței software. Cu toate acestea anterior definirii setului de criterii se identifică caracteristicile

esențiale ale aplicației de optimizat. Principalele criterii şi nivelurile lor de optim sunt:
- minimizarea volumului de prelucrări, duratei unei tranzacții, complexității programului, numărului de iterații, zonei de memorie pentru rezultate intermediare;
- maximizarea dimensiunii problemei, fiabilității, gradului de utilizare a memoriei rezervate, preciziei, gradului de flexibilitate, gradului de mentenanță;

În procesul de optimizare, dacă se cunoaşte un anumit obiectiv este necesar să se studieze dacă alte obiective au tendințe contradictorii.
De exemplu creşterea fiabilității conduce la creşterea complexității. Secvența de program care realizează raportul a două numere este implementată în cel mai simplu mod prin intermediul instrucțiunii: e = a/b; expresie care are complexitatea $C_1 = 3 \log_2 3 + 2 \log_2 2$. Creşterea gradului de fiabilitate implică implementarea secvenței:

```
vb = 0;
if (b != 0){ e = a/b; vb = 1;}
```

Această expresie are complexitatea egală cu $C_2 = 8 \log_2 + 6 \log_2 6$. Deci $C_2 > C_1$.
Se maximizează fiabilitatea dar complexitatea este cu sens de a fi maximizată, ori ea trebuie să scadă.
Optimizarea prin caracterul ei cuprinzător generează efecte combinate asupra programului. Dacă lungimea programului este măsurată ca număr de instrucțiuni L, pentru programul P, din figura 2, ce execută apeluri de subprograme se identifică apelurile distincte de subprograme.

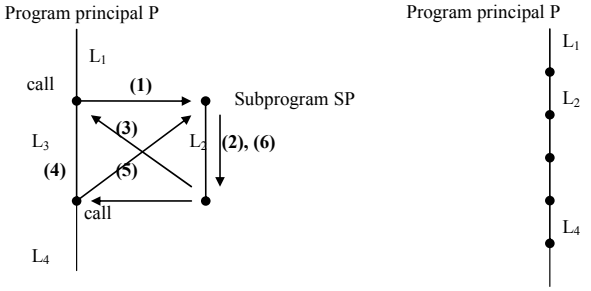

Fig. 2. Apeluri de subprograme

Lungime acestui program ca număr de apeluri de subprogram este $LgP = L_1 + L_2 + L_3 + L_4$.
Utilizarea de subprograme presupune instrucțiuni de salt necondiționat în segmente diferite care cresc numărul de cicluri maşină. De aceea se cere incorporarea procedurilor în program atunci când se apelează de foarte multe ori, figura 2. Lungime secvenței de subprograme este $Lg_2\ P = L_1 + L_2 + L_3 + L_2 + L_4$.
Într-adevăr $Lg_1 < Lg_2$ dar nu mai apar salturi necondiționate.

Construirea variantelor de program

O problemă P se construieşte în numeroase variante funcție de limbajul de programare utilizat, tehnici de programare, structuri de date folosite, mod de abordare: distribuit, rețea, standalone.
Pentru fiecare variantă intervin elemente de optimizare. După aceea se pune problema alegerii variantei celei mai potrivite, concept care se identifică în mare măsură cu procesul de optimizare.
Există nenumărate modele de analiză, proiectare şi implementare a unui produs software. Marea majoritate urmăresc realizarea unei variante funcționale care este îmbunătățită într-un proces repetitiv până când se ating nivelurile stabilite ca obiective finale sau până când raportul calitate / cost este optim.
Fiecare dintre variantele produsului, cu excepția prototipului inițial, se obține în urma testării şi evaluării variantei precedente, care este supusă modificărilor la niveluri diferite, cod sursă, framework, algoritmi, pentru a fi optimizată. Îmbunătățirea calității variantelor produsului program este un proces bazat pe optimizare entropică.
Efectele optimizării sunt materializate prin creşterea calității produsului program. Există situații în care se alege implementarea soluției economice în defavoarea soluției performante, însă aceste decizii privesc părți specifice ale aplicației şi sunt luate în considerare dacă nu afectează descrierea produsului în ansamblul său.
Optimizarea şi metodele de realizare a acesteia capătă sensuri diferite în realizarea variantelor de program funcție de obiectivele urmărite. Din perspectiva entității produsului software, optimizarea conduce la obținerea variantei finale, care este şi cea mai bună din

punct de vedere al producătorului.
Realizarea variantelor de program presupune obținerea de produse software funcționale intermediare care să reprezinte baza de evaluare a efectelor optimizării multicriteriale sau unicriteriale prin măsurarea nivelurilor caracteristicilor analizate. Obținerea nivelurilor agregate permite derularea analizei comparate a variantelor de program care să conducă la alegerea soluției optime.
Se consideră lista formată din şapte variante a produsului program *Prog*, $Prog_1$,$Prog_2$, $Prog_3$, $Prog_4$, $Prog_5$, $Prog_6$, $Prog_7$ ce implementează în moduri variate metode identice necesare rezolvării problemei *Prob*. Obiectivul analizei constă în determinarea soluției optime, reprezentată de una dintre variante care să rezolve problema *Prob*, astfel încât să fie obținute cele mai bune niveluri pentru caracteristicile de calitate software considerate.
Lista de produse software se consideră omogenă prin prisma:
- setului de date de intrare ce are caracteristici comune pentru toate programele;
- problemei de rezolvat;
- platformei hardware şi software pe care rulează aplicațiile analizate.

Diferențele dintre programe au la bază modul de implementare în surse a metodei de rezolvare. Algoritmul de prelucrare a datelor diferă prin modalitatea de reprezentare a modelului de date şi prin complexitatea metodei de obținere a rezultatelor.
Se consideră setul de caracteristici de calitate software *SQCS* : C_1 - durata unei tranzacții; C_2 – durata de prelucrare; C_3 – grad de încărcare a setului de date de intrare; C_4 – volum de prelucrări.
Setul de caracteristici variază în funcție de obiectivul final al analizei şi reprezintă un instrument pus la dispoziția cercetătorului. Prin intermediul caracteristicilor şi a metricilor software execuția programului analizat este descrisă de valori numerice ce reprezintă niveluri reale, măsurate. Validarea procesului de evaluare a produsului software evidențiază semnificația măsurătorilor şi atestă faptul că niveluri obținute pentru aplicații software diferite sunt comparabile.
Tabelul 1 descrie valorile medii obținute pentru caracteristicile analizate, în urma rulării programelor cu 30 seturi de date diferite.

Tabel 1. Nivelurile medii măsurate pentru caracteristicile analizate.

Caracteristică	Prog1	Prog2	Prog3	Prog4	Prog5	Prog6	Prog7
C_1	1.87	2.33	2.87	1.47	1.93	3	2.83
C_2	128.3667	125.2667	148.9667	236.9	154.7333	139	164.7667
C_3	0.528333	0.557667	0.511	0.454333	0.500667	0.455667	0.593333
C_4	2822.4	3528	4334.4	2217.6	2923.2	4536	4284

Obținerea unui indicator agregat construit pe baza acestor niveluri direct măsurate porneşte de la identificarea coeficienților de importantă asociați caracteristicilor analizate. Obținerea variantei optime de program care să rezolve problema *Prob* analizează influențele tuturor caracteristicilor software luate în discuție. În acest sens, coeficientul de importantă ic_k, asociat caracteristicii C_k, descrie ponderea acesteia în caracterizarea produsului software ca fiind optim.
Pe baza analizelor anterioare şi prin studiul dependenței dintre caracteristicile C_1, C_2, C_3, C_4 şi viteza de obținere a rezultatului s-au identificat valorile coeficienților de importanță din tabelul 2 aşa cum rezultă din punctele acordate de un lot de 30 de specialişti.

Tabel 2. Coeficienți de importanță pentru caracteristicile analizate.

Caracteristici	Ponderi	Tip
C1 – durată tranzacție	0.47	Minim
C2 – durată prelucrare	0.32	Minim
C3 - grad încărcare	0.09	Maxim
C4 - volum prelucrări	0.12	Minim
Total	1	-

Fiecare caracteristică software analizată are asociată o funcție de optimizare care să conducă la alegerea nivelului eficient. Identificarea acestei funcții are importanță în compararea directă a nivelurilor caracteristicii pentru două sau mai multe variante de program. Tabelul 2 descrie tipul funcțiilor asociate criteriilor C_1, C_2, C_3 şi C_4.

Tabel 3. Nivelurile maxime şi minime asociate criteriilor.

Criterii	MIN	MAX	Amplitudinea = MAX - MIN
C1 - durate	1.466667	3	1.533333
C2 - costuri	125.2667	236.9	111.6333
C3 - grad inc	0.454333	0.593333	0.139
C4 - vol prel	2217.6	4536	2318.4

Reprezentând criterii diferite se impune normalizarea valorilor în vederea determinării unui indicator agregat care să descrie varianta de produs software printr-o valoarea direct comparabilă cu valorile agregate asociate restului de variante. Dacă criteriul C_i are funcție de optimizare de tip *Minim*, valoarea normalizată se determină folosind prima relație, în caz contrar se utilizează ce-a de-a doua expresie: $VNC_i^j = \frac{VC_i^{\max} - VC_i^j}{VC_i^{\max} - VC_i^{\min}}$ (1)

$$VNC_i^j = \frac{VC_i^j - VC_i^{\min}}{VC_i^{\max} - VC_i^{\min}} \quad (2)$$

în care:

VNC_i^j – valoarea normalizată a criteriului C_i pentru programul $Prog_j$;

$VC_i^{\max}$ – valoarea maxim înregistrată de criteriul C_i pentru variantele testate;

$VC_i^{\min}$ – valoarea minim înregistrată de criteriul C_i pentru variantele testate;

VC_i^j – valoarea înregistrată de criteriul C_i pentru varianta curentă de program $Prog_j$.

Tabelul 4 descrie valorile normalizate ale criteriilor C_1, C_2, C_3, C_4 pentru fiecare variantă de program.

Tabel 4. Valori normalizate.

Criterii	Prog1	Prog2	Prog3	Prog4	Prog5	Prog6	Prog7
C1 - durate	0.73913	0.434783	0.086957	1	0.695652	0	0.108696
C2 - costuri	0.972231	1	0.787698	0	0.736041	0.876978	0.646163
C3 - grad inc	0.532374	0.743405	0.407674	0	0.333333	0.009592	1
C4 - vol prel	0.73913	0.434783	0.086957	1	0.695652	0	0.108696
Indicator agregat	**0.795114**	**0.643428**	**0.340058**	**0.59**	**0.675968**	**0.281496**	**0.360903**

Valoare indicatorului agregat IA este determinată prin ponderarea valorilor normalizate cu valoarea coeficientului de importanță asociat fiecărui criteriu prin relația

$$IA^j = \sum_{i=1}^{4} VNC_i^j * p_i$$

Alegerea variantei optime de program este condiționată de o valoare mare a indicatorului agregat. Deci prima variantă este identificată ca fiind soluția optimă de rezolvare a problemei *Prob*.

Evaluarea efectelor presupune testare şi construire date de test. De exemplu realizarea variantelor de program care sortează un şir prin diferite metode. Varianta ce implementează algoritmul metodei de sortare prin interschimbare este testată pornind de un set de date gata sortat, care apoi este modificat treptat până la transformarea într-un şir sortat în sens opus. Varianta de sortare prin selecție este testată în aceleaşi condiții.

Pornind de la acelaşi sistem de date este permisă verificarea numărul de comparări şi realizarea unei analize comparate. Pentru seturi de date diferite se măsoară legătura dintre lungime set de date şi număr de comparări.

De asemenea, pentru optimizare se alege varianta de utilizare a unei structuri de date care să ofere prin construcția sa o modalitate de găsire rapidă a informațiilor. În acest sens se

evaluează implementarea unui arbore binar de căutare prin măsurarea duratei şi a numărului de comparări necesare găsirii elementelor.

Printr-un proces de analiză comparată obiectiv şi complet se alege varianta cea mai potrivită.

Un indicator utilizat în evaluarea efectelor optimizării şi care oferă o imagine cantitativă a acestui proces este volumul de prelucrări ca număr de cicluri maşină, dat de relația $V = \sum_{k=1}^{n} nc(I_k)$, în care $nc(I_k)$ este numărul de cicluri maşină ale instrucțiunii I_k.

Optimizare multicriterială

Complexitatea aplicațiilor informatice impune luarea în considerare în cadrul procesului de optimizare a mai multor caracteristici analizate. Îmbunătățirea programului urmăreşte nivelurile uneia sau mai multor caracteristici de calitate software. Optimizarea unicriterială măsoară şi compară nivelul caracteristicii, QC, analizate pentru toate variantele de program. Cu toate acestea, modificarea sursei programului având ca obiectiv îmbunătățirea criteriului QC implică modificări pentru o mulțime de caracteristici direct sau indirect dependente.

Procesul de optimizare multicriterială ia în considerare un set de k caracteristici de calitate software, $QC_1, QC_2, ..., QC_k$, pentru a decide care variantă de program reprezintă modalitatea optimă de rezolvare a problemei. Neomogenitatea setului de caracteristici din punctul de vedere al semnificației valorilor impune definirea unui indicator care să reprezintă valoarea agregată a celor k niveluri direct măsurate. Indicatorul IQC se calculează pentru fiecare variantă de program şi are la baza determinarea anterioară a ponderilor de importanță asociate fiecărei caracteristici QC_j, cu $j = 1..k$.

Alegerea setului de caracteristici software rezultă din obiectivul stabilit de producător pentru a obține cel mai bun program şi nu dintr-o analiză anterioară a relațiilor dintre diferite caracteristici de calitate software.

Se consideră variantele de program Prg_1, Prg_2, …, Prg_{nprg} şi setul de caracteristici software $QC_1, QC_2, ..., QC_k$ în funcție de care se alege soluția optimă. Fiecare dintre programele Prg_i, cu i=1..nprg, sunt analizate măsurându-se nivelul qcl_{ij} caracteristicilor QC_j, cu $j = 1..k$.

Caracteristicile de calitate software diferă între ele prin:

- semnificația valorii direct măsurate;
- criteriul de optim; din acest punct de vedere caracteristicile de calitate formează două mulțimi pentru care criteriul de optim este unul de minim, respectiv de maxim;
- metrica utilizată în măsurarea nivelului;
- ponderea importanței caracteristicii pentru o categorie de produse software.

Luarea în considerare a criteriului de optim este impus de determinarea ulterioară a indicatorului agregat ICQ care trebuie să țină seama de faptul că unele dintre caracteristicile QC_j au punct de optim o valoare minimă, respectiv maximă. Valorile normalizate pentru nivelurile măsurate sunt obținute prin transformările:

$$qcl'_{ij} = \begin{cases} \min\{qcl_{i1}, qcl_{i2}, \ldots, qcl_{ij}, \ldots, qcl_{ik}\} - qcl_{ij}, \text{ cu } i = 1..nprg, \text{ dacă punctul} \\ \qquad \text{de optim al caracteristicii } QC_j \text{ este unul de minim;} \\ qcl_{ij} - \min\{qcl_{i1}, qcl_{i2}, \ldots, qcl_{ij}, \ldots, qcl_{ik}\}, \text{ cu } i = 1..nprg, \text{ dacă punctul} \\ \qquad \text{de optim al caracteristicii } QC_j \text{ este unul de maxim;} \end{cases}$$

Determinarea soluției optime se bazează pe calcularea valorii indicatorului IQC_i pentru fiecare program Prg_i prin intermediul relației $IQC_i = \sum_{j=1}^{k} qcl'_{ij} * qcic_j$ în care $qcic_j$ reprezintă coeficientul de importanță asociat caracteristicii de calitate software QC_j, cu $j=1..k$. Determinarea ponderilor reprezintă un proces separat care descrie importanța fiecărei caracteristici software QC_j în imaginea calitativă de ansamblu a soluției Prg_i, cu $i = 1..nprg$.

Soluția optimă este identificată prin compara-

rea directă a valorilor aferente indicatorului IQC_i. Varianta de program Prg_{opt} pentru care este adevărată relația:

$IQC_{opt} < IQC_i$ cu $i = 1..nprg$ şi $i \neq opt$

În cazul în care există două sau mai multe soluții optime, se alegea varianta pentru care caracteristica cu coeficientul de importanță cel mai mare are cel mai bun nivel înregistrat. Se consideră trei soluții Prg_1, Prg_2, Prg_3 pentru rezolvarea problemei de determinare a sumelor elementelor de pe fiecare coloană a unei matrice rară memorată într-un fişier. Alegerea variantei optime de program se face în funcție de caracteristicile: QC_1 – Viteza de prelucrare; QC_2 – Spațiu ocupat; QC_3 – Complexitate program QC_4 – Fiabilitate; ale căror ponderi de importanță sunt descrise în tabelul 5.

Tabel 5. Ponderile de importanţă ale caracteristicilor.

CARACTERISTICĂ	QC_1	QC_2	QC_3	QC_4
COEFICIENT DE IMPORTANȚĂ (qcic)	34%	18%	28%	20%

Soluțiile diferă între ele prin implementarea de condiții suplimentare de validare a datelor de intrare, algoritmi diferiți pentru calcularea sumelor, modalități distincte de citire a datelor din fişier, structuri de date variate.

În urma rulării celor trei variante de program se obține pentru setul de caracteristici analizate valorile din tabelul 6.

Tabel 6. Nivelurile caracteristicilor de calitate software.

	CARACTERISTICĂ DE CALITATE SOFTWARE			
VARIANTĂ DE PROGRAM	**QC_1**	**QC_2**	**QC_3**	**QC_4**
Prg_1	**3,12**	**12,00**	**9,34**	**3,00**
Prg_2	**3,82**	**17,00**	**19,56**	**9,00**
Prg_3	**3,52**	**19,00**	**15,61**	**8,00**
Punct de optim	**minim**	**minim**	**minim**	**maxim**
Valoare optimă	**3,12**	**12,00**	**9,34**	**9,00**

În vederea determinării indicatorului agregat IQC sunt normalizate valorile din tabelul 6. Datele din tabelul 7 reprezintă fundamentul procesului de alegere a soluției optime.

Tabel 7. Valorile normalizate ale caracteristicilor de calitate software analizate.

	CARACTERISTICĂ DE CALITATE SOFTWARE				VALOAREA
VARIANTĂ DE PROGRAM	**QC_1**	**QC_2**	**QC_3**	**QC_4**	**ICQ**
Prg_1	**0,00**	**0,00**	**0,00**	**-6,00**	**-1,2**
Prg_2	**-0,70**	**-5,00**	**-10,22**	**0,00**	**-3,9996**
Prg_3	**-0,40**	**-7,00**	**-6,27**	**-1,00**	**-3,3516**

Odată determinate valorile indicatorilor IQC din tabelul 7 se identifică varianta de program optimă ca fiind aceea pentru care indicatorul IQC are valoare minimă. Se observă că varianta de program Prg_1 conduce la obținere nivelurilor optime.

Rezultatele măsurătorilor sunt utilizate ulterior la evaluarea influenței pe care o are dependența dintre caracteristicile de calitate software asupra obținerii unei soluții optime. Acest lucru se realizează prin analiza suplimentară a relației dintre caracteristici şi prin construirea de modele de regresie care să includă acele caracteristici între care există o relație puternică.

Determinarea unui model matematic prin intermediul căruia se determină valoarea indicatorului agregat are la bază analiza dependenței dintre caracteristicile software de optimizat. Pe baza analizei empirice a unui set omogen de produse program care reprezintă variante de implementare a metodei de prelucrare a datelor sunt identificate ponderile de importanță pentru fiecare criteriu în parte.

Absența studiului empiric şi orientarea pur teoretică a cercetării conduce la obținerea de rezultate care sunt lipsite de semnificație şi care nu sunt validate de practică.

Concluzii

Optimizarea reprezintă un proces continuu care se finalizează odată cu decizia de a înlocui aplicația software cu o alta prin reluarea

activităţilor de analiză, proiectare şi implementare.
Dezvoltarea permanentă şi cu un ritm mult prea rapid a societăţii informaţionale prin lansarea de noi tehnologii şi instrumente de realizare a produselor software susţine caracterul de continuitate al optimizării oferind mijloacele necesare atingerii criteriilor de optim.
Abordarea empirică fundamentează reguli practice de dezvoltare software. Utilizarea acestor reguli creşte eficienţa produselor software.

Bibliografie

[ALLE01] Eric E. Allen – *Diagnosing Java Code: Improve the performance of your Java code*, Java Developerworks, http://www106.ibm.com/developerworks/java/library/j-diag8.html
[BOJA05] Cătălin BOJA – *Software multicriterial optimization*, The proceedings of the 7th International Conference on Informatics in Economy, May 2005, Academy of Economic Studies, Bucharest, Romania, 2005.
[CURTI99] Petru CURTICAPEAN – *Metode de cuantificare a efectelor optimizării sistemelor de programe*, Referat prezentat în cadrul pregătirii Tezei de Doctorat, ASE, Bucureşti, 1999.
[DUMI04] Eugen DUMITRASCU – *Algoritmi de optimizare a produselor software*, Referat prezentat în cadrul pregătirii Tezei de Doctorat, ASE, Bucureşti, 2004.
[FOG99] Agner FOG – *How to optimize for the Pentium family of microprocessors*, Copyright by Agner Fog, 1999.
[IVAN83] Ion IVAN, S. COMAN, Al. BALOG, R. ARHIRE – *Tehnici de evaluare a efectelor optimizării de programe*, Revista de statistică, nr. 3, 7, 1983.
[IVAN96] *Ion IVAN, Cristian CODREANU – Optimizarea programelor assembler, Raport de cercetare, Bucureşti, 1996.*