UNIVERSITATEA POLITEHNICA BUCURESTI

**METRICI SOFTWARE**

**Broasca Valentina-Giorgiana**

**GRUPA 441A**

**CUPRINS**

**1. Introducere**

 **1.1 Definitie**

 **1.2 Nevoia de metrici software**

 **1.3 Clasificarea metricilor software**

 **1.4 Complexitatea metricilor software**

 **1.5 Proprietatile metricilor software**

 **1.6 Modelul de calitate**

**2. Complexitatea Halstead**

 **2.1 Descriere**

 **2.2 Avantaje**

 **2.3 Dezavantaje**

 **2.4 Exemplu**

**3. Linii de cod LOC**

 **3.1Metoda evaluarii dimensionale**

 **3.2Evaluarea functionala**

 **3.2Tehnici de decompozitie**

 **3.3Exemplu**

**4. Bibliografie**

 **1.INTRODUCERE**

 Aprecierea unui sitem de programe se face in mod calitativ (bun, satisfacator, foarte bun, nesatisfacator)si in mod cantitativ (numeric). Pentru a face aceasta metricile software inglobeaza modele, indicatori si proprietatile acestora, precum si modalitati de evaluare si validare.

 Complexitatea software este o caracteristică de calitate care se regăseste in calculul devizelor pentru dezvoltarea software si pentru corectarea indicatorului de productivitate a programatorilor.

 Metricile software sau definit pe baza a trei surse: metrici bazate pe textul sursă, metrici bazate pe graful asociat programului, metrici de comportament care inregistrează niveluri ale parametrilor in timpul executiei programului.

 Deci pentru a gestiona eficient un proces, acesta necesită cuantification, measurement, and modeling.tificare, măsurare, si modelare. Software Metricile software oferă o bază cantitativă pentru dezvoltareaacceptance. ment and validation of models of the software devel-si validarea modelelor procesului de dezvoltare software. Metrics can be used to improve Metricile sunt utilizate pentru imbunătătireaFaced with this situation, the author has chosen to software productivity and quality. productivitatii si calitatii software. This module in- and reviews their use in constructing models of the Although current

 Aceasta prezentare isi propune sa prezinte concepte de baza, modele si modalitati de utilizare a metricilor software.

1.Definitia Metricii Software

 Pentru a intelege mai bine acest concept, este important sa definim in continuare *software-ul.* Essentially, *software* Software-ul *metrics* deals with the measurement of the software se ocupă cu *măsurătorile* produsului software part of a larger area that might be called *software*product and the process by which it is developed. si a procesului prin care acesta este dezvoltat. In Astfel this discussion, the software product should be, produsul software ar trebui să fie confine itself to software metrics as defined above.viewed as an abstract object that evolves from an privit ca un obiect abstract care evoluează de la o initial statement of need to a finished software sys- declaratie initială de necesitate la un sistem software terminat tem, including source and object code and the, inclusiv codul sursă si codul obiect si various forms of documentation produced during de- diverse forme de documentare produse in timpul de velopment. dezvoltare.

 Ordinarily, these measurements of the In mod obisnuit, aceste sunt studiate si dezvoloped for use in modeling the software developmenttate pentru utilizarea in modelarea de procese software. process. These metrics and models are then used to Aceste valori si modele sunt apoi folosite pentru a estimate/predict product costs and schedules and to estima / anticipa costurile produselor si să (either source or object code), or the number of measure productivity and product quality. măsoare productivitatea si calitatea produselor. Informa- Informatiile tion gained from the metrics and the model can then obtinute din măsurători apoi pot fi utilizate in gestionarea si controlul dezvoltarii • software of poor quality, andopment process, leading, one hopes, to improvedprocesului de imbunătătire a results. rezultatelor. Good metrics should facilitate the development of

 Măsurătorile bune ar trebui să faciliteze dezvoltarea de . models that are capable of predicting process or modelele care sunt capabile de a prezice un proces sau product parameters, not just describing them. parametrii de produs, nu doar a le descrie. Thus, Astfel, metricele ideale ar trebui sa fie simple si precis definite, astfel incat să fie

clear how the metric can be evaluated; clar cum o metrica poate fi evaluată:

* • objective, to the greatest extent possible; obiectiv, in cea mai mare măsură posibilă; products, and higher productivity.• easily obtainable ( *ie* , at reasonable cost);
* cu un cost rezonabil
* usor de obtinut
* • valid—the metric should measure what itmetrica ar trebui să măsoare ceea ce is intended to measure; and este destinată să măsoare
* • robust—relatively insensitive to (intuitive- insensibila la ly) insignificant changes in the process or modificări nesemnificative in procesul de dezvoltare al produsului product.. In addition, for maximum utility in analytic studies

 In plus, pentru utilitate maximă in studiile analitice and statistical analyses, metrics should have data si analizele statistice, măsurătorile ar trebui să aibă values that belong to appropriate measurement valori care apartin de măsuratoarea corespunzătoare.

.scales

. . Calităti fundamentale required of any technical system are necesare pentru orice sistem tehnic sunt:

* • functionality—correctness, reliability, etc.; functionalitate,corectitudine, fiabilitate
* • performance—response time, throughput, performantă, timp de răspuns, debit, speed, etc.; andviteza
* • economy—cost effectiveness. Cost scazut, eficacitate.

So far as this author can discern, *software metrics* ,

 Performanta este cu sigurantă importanta, dar nu este inclusa in cadrul discutiilor de software, metrics, except regarding whether the product meets cu o exceptie in ceea ce priveste dacă produsul indeplineste software metrics cannot meaningfully be discussed specific performance requirements for that product.cerintele specifice de performantă pentru acest produs.

 Evaluarea performantei este deseori tratataThe evaluation of performance is often treated ex- de către cei implicati in *evaluarii de performantă,* *tion* studies, but these are not generally included in studii *de renovare,* dar acestea nu sunt in general incluse in tion of the software crisis. what is referred to as *software metrics*ceea ce este cunoscut ca *metrica software*

.

 Este posibil ca, in viitor, domeniul de aplicare al *metricilor* *software* It is important to further define the term *softwaremetrics* may be expanded to include performance sa fie extins pentru a include performanta evaluation, or that both activities may be considered si evaluarea, sau ca ambele activităti sa poata fi considerate part of a larger area that might be called *software* parte dintr-o suprafată mai mare care ar putea fi numit *software* *measurement* . *de măsurare.*

2.Nevoia de Metrici software

 Criza software trebuie să fie abordata si, in extent possible, resolved. măsura in care este posibil, rezolvata. To do so requires more Pentru a face acest lucru se necesită mai multe accurate schedule and cost estimates, better quality programe exacte si estimări de costuri, calitate mai bună products, and higher productivity. pentru produse, si productivitate mai mare. All these can be Toate acestea pot fi achieved through more effective software manage- realizate printr-un management software mai eficient , care, la randul său, poate fi facilitat prin utilizarea de metrici software.

 Current software Managementul software curent este ineficient, deoarece dezvoltarea software• robust—relatively insensitive to (intuitive-opment is extremely complex, and we have few este extrem de complexa, si avem doar catevaly) insignificant changes in the process or well-defined, reliable measures of either the processmăsuri de incredere bine definite din procese or the product to guide and evaluate development.sau produse pentru a indruma si evalua dezvoltarea. Thus, accurate and effective estimating, planning,Estimarea eficienta si exacta, planificarea, and control are nearly impossible to achievesi controlul sunt aproape imposibil de realizat .. Improvement of the management process de- Imbunătătirea procesului de gestionare a pends upon improved ability to identify, measure, lucrului depinde de abilitatea imbunătătită de a identifica, măsura, and control essential parameters of the development si controla parametrii esentiali ai dezvoltării process. proceselor. This is the goal of software metrics—the Acesta este scopul indicatorilor Software. :identification and measurement of the essentialIdentificarea si măsurarea sunt esentialii parameters that affect software development. parametri care afectează dezvoltarea de software. Software metrics and models have been proposed

 Metricile software si modele au fost propuse speed, etc.; andand used for some time si folosite de ceva timp. . Metrics, however, have rarely been used in any reg-Rezultatele recente indică as the term is most commonly used today, concerns that the conscientious implementation and applica-că punerea in aplicare a unei combinatii intre un program software si metrici poate ajuta la atingerea better management results, both in the short run (forunor rezultate mai bune, atat pe termen scurt (pentru a given project) and in the long run (improving un anumit proiect) cat si pe termen lung (imbunătătirea productivity on future projects) productivitatii pentru proiecte viitoare). metrics, except regarding whether the product meets software metrics cannot meaningfully be discussedO mai bună utilizare The evaluation of performance is often treated ex- of existing metrics and development of improveaasssssaa metricilor existente pare a fi un factor important in rezolvarea crizei software-ului.

3.Clasificarea metricilor

Metricile software pot fi clasificate in mare ca metrici de produs sau de proces, mai poat fi si de resurse.

Metricile de produs sunt masuratori ale produsului software in orice stadiu de dezvoltare, de la cerinte pana la sistemult instalat. Acest tip de metrici pot masura complexitatea design-ului software-ului, dimensiunea programului final sau numarul de pagini de documentatie produs.

Matricele de proces, pe de alta parte, sunt masuratori ale procesului de dezvoltare a software-ului, asa ca timpul de dezvoltare total al programului, tipul de metodologie folosit, sau gradul de experienta al programatorilor.

Metricile mai pot fi clasificate ca obiective sau subiective. In general, metricile obiective ar trebui sa dea acelasi rezultat indiferent de observator, pe cand metricile subiective pot duce la rezultate diferite deoarece intervine judecata celui care face masuratorile.

Ca si comparatie, putem spune ca:

* Metricile de proces
	+ Conduc la imbunatatirea produsului pe termen lung
* Metricile de produs
	+ Ofera o imagine asupra stadiului proiectului
	+ Urmareste potentiale riscuri
	+ Descopera zone cu probleme
	+ Regleaza fluxul de lucru si sarcinile
	+ Evalueaza abilitatea echipei de a controla calitatea

Tipuri de masuratori:

* Masuratori directe (atribute interne)
	+ Cost, efort, LOC, viteza, memorie
* Masuratori indirecte (atribute externe)
	+ Functionalitate, calitate, complexitate, eficienta, fiabilitate si mentenabilitate

 4.Complexitatea metricilor

 Definirea de metrici software revine la a construi modele, indicatori care pornesc de la marimi ce se masoara cu obiectivitate (numar linii sursa, numar erori, numar variabile, numar instructiuni de Intrare/Iesire, numar functii, numar parametrii transmisi, numar nivele ale arborelui asociat).

 Practica arata ca exista o stransa legatura intre comportamentul efectiv al unui program si structura lui exprimata prin marimi ce se determina exact, obiectiv.

 Intradevar, masurarea complexitatii care poate fi calculata mai devreme in dezvoltarea programului, ar fi de mai mult ajutor in manevrarea procesului software. Masuratorile lui McCabe se bazeaza pe forma finala a codului programului.

**Complexitatea ciclomatica – v(G)**

Luand ca exemplu orice program, ii putem desena graficul fluxului de control G, in care fiecarui nod ii corespunde unui bloc secvential de cod, si fiecare arc ii corespunde unei ramuri sau unui punct de decizie in program. Formula unui astefl de graf poate fi: v(G)=e-n+2 , unde e este numarul de laturi, iar n este numarul de noduri din graf. V(G) poate fi folosit pentru a masura complexitatea unui program, si poate fi un bun ghid in dezvoltarea si testarea produsului.

**Noduri**

Un nod este definit ca o trecere necesara pentru liniile directionale in graf. Numarul de noduri dintr-un program tine practic de complexitatea programului.

**Fluxul de informatie**

Fluxul de informatie poate fi deasemenea folosit ca parametru de masurare a complexitatii programului. Aceasta metoda presupune practic numararea de fluxuri de informatie locale care intra (fan-in) si care ies (fan-out) din fiecare procedura. Astfel complexitatea procedurii poate fi definita ca: c=[lungimea procedurii]\*[fan-in\*fan-out]2 . Dezavantajul acestei proceduri este ca poate da ca rezultat complexitate 0 unui program daca acesta nu are legaturi externe.

**Exemplu:**

i = 0;

while (i<n-1) do

 j = i + 1;

 while (j<n) do

 if A[i]<A[j] then

 swap(A[i], A[j]);

 end do;

 i=i+1;

end do;

Graful de control:

Unde :

5.Proprietatile metricilor

Putem pune in evidenta proprietati ale metricilor softaware, dupa cum urmeaza:

 P1 : o metrica trebuie sa puna in evidenta doua subclase de programe : programe utilizabile si programe inutilizabile;

 P2 : doua programe identice conduc la valori egale ale unei metrici care le evalueaza;

 P3 : utilizarea unei metrici nu conduce la situatii anormale cand se fac studii empirice;

 P4 : daca o metrica deriva dintr-o alta metrica, apartenentele programelor la subclase nu se modifica;

 P5 : modificarilor neutre din program nu le corespund cresteri sau diminuari ale valorilor rezultate dintr-o metrica;

 P6 : modificarilor limitate din program le corespund cresteri sau diminuari limitate ale valorilor rezultat dintr-o metrica;

 P7 : cresterea numarului de variabile exogene conduce la cresterea gradului de masurabiliate asociat metricii;

 P8 : modificarilor active din program trebuie sa le corespunda diminuari sau cresteri ale valorilor rezultate dintr-o metrica.

 Proprietatile metricilor sunt puse in evidenta prin parcurgerea unor pasi si anume :

 - construirea unui sistem al specificatiilor;

 - definirea ierarhiilor asociate modulelor din care este format programul;

 - se calculeaza pentru fiecare modul o serie de indicatori ( pe text, structura sau dinamica);

 - indicatorii calculati se agrega rezultand pentru fiecare modul un indicator agregat.

6.Modelul de calitate



Imag. 1[1]



Imag.2 [2]

#  2. COMPLEXITATEA HALSTEAD

 Complexitatea software este o caracteristică de calitate care se regăseste in calculul devizelor pentru dezvoltarea software si pentru corectarea indicatorului de producti-vitate a programatorilor.

 Metricile software s-au definit pe baza a trei surse: metrici bazate pe textul sursă, metrici bazate pe graful asociat programu-lui, metrici de comportament care inre-gistrează niveluri ale parametrilor in tim-pul executiei programului.

 Metrica Halstead este definita prin indicatorii:

C = complexitatea programului

E = efortul de programare

V = volumul programului

L = nivelul programului

 unde:



sau, considerand  si   putem scrie:



 

avand aceeasi semnificatie ca si , dar contorizează totalurile, nu numai pe cele distincte.



V\* este volumul minim al programului, care este calculat din numărul minim de parametrii I/O necesar pentru a specifica operatia unui algoritm si returul rezultatului, avand expresia:

  ,

iar este numărul de parametri de I/O folositi in apelul programului.



Ecuatia de timp:

T\*=(n1N2(n1log2n1+n2log2n2)log2n)/2n2S

cu:

n1 – numărul de tipuri fundamentale de date care apar distinct in program

n2 – numărul de tipuri derivate de date care apar distinct in program

n3 – numărul de instructiuni distincte utilizate de programator

n4 – numărul de operanzi distincti care apar in program

n5 – numărul de operatori distincti pentru referire care apar in program

n6 – numărul de functii distincte apelate

 Metrica Halstead are ca obiectiv masurarea urmatoarelor atribute ale unui program:

-lungimea

-vocabularul

-volumul

-nivelul de dificultate

-nivelul de inteligenta al continutului

-efortul de programare

-timpul necesar pentru programare

 Metricile Halstead se bazeaza pe identificarea si numararea a doua tipuri de atribute din program: numarul de operanzi si numarul de operatori.

1.Avantaje ale metricilor Halstead:

-nu necesita o analiza aprofundata a structuri programului

-prezice rata erorilor

-prezice efortul de intretinere

-este folositoare in a programa si raporta proiecte

-masoara calitatea totala a programelor

-este simplu de calculat

-poate fi folosita pentru orice limbaj de programare

2.Dezavantaje ale metricilor Halstead:

-Depinde de codul complet

-Este nefolositor ca si model estimativ predictiv. Modelul McCabe este mai potrivit cand vine vorba de design.

3.Exemplu:

**I.**

void sort ( int \*a, int n ) {

int i, j, t;

if( n<2 )

return;

for( i=0 ; i<n-1; i++){

for( j=i+1 ; j<n ; j++){

if( a[i] >a[j]){

t=a[i];

a[i] =a[j];

a[j] =t;

}}}}

Daca ignoram functiile si numaram operanzi si operatorii, obtinem:

Operanzi: 1 0 2 1 1 2 6 a 8 I 7 j 3 n 3 t

Operatori: 3 5 1 1 2 9 4 4 6 < = > - , ; ( ) [ ] 3 3 1 2 2 2 1 1 + ++ for if int return { }

In total avem N1=50 operatori din care n1=17 unici, si N2=30 operanzi din care n2=7 unici.

Volumul: V=80 log2(24) = 392 apartine intervalului [20,1000]

Nivelul de dificultate: D=(n1/2)\*(N2/n2) = 17/2\*30/7 = 36

Efortul: E=V\*D= 392\*36 = 14112

Timpul de implementare: T=E/18= 784 sec, aproximativ 13 min

Erori: E2/3/3000 = 0.19

**II.**

01 public class Class1 {

02 public void x(boolean v) {

03 int i;

04 if (v) {i = 1;}

05 else {i = 2;}

06 switch(i){

07 case 1:

08 case 2:

09 default:;

10 }

11 try {

12 while(v){

13 v = false;

14 int r = 1;

15 Boolean b = new Boolean(true);

16 i = i\*i+r;

17 break;

18 }

19 } catch (Exception e) {

20 throw e;

21 }

22 }

23 }

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Linia de cod**  | **N1**  | **n1**  | **N2**  | **n2**  |
| 04  | if, = | if, = | v, i, 1 | v, i, 1 |
| 05 | = |  | i, 2 | 2 |
| 06 | switch | switch | i  |  |
| 12 | loop | loop | v |  |
| 13 | = |  | v, false | false |
| 14 | = |  | 1 |  |
| 15 | =,new | new | true | true |
| 16 | =, \*, + | \*, + | i, i, i, r  | r |
| 20 | throw | throw | e | e |
| **Total**  | **13**  | **8**  | **16**  | **8**  |

Stiind ca: n1=prima aparitie a operatorilor

 N1=de cate ori se repeta acel operator

 n2=prima aparitie a operanzilor

 N2=de cate ori se repeta acel operand,

putem calcula vocabularul programului, lungimea programului, volumul, nivelul de dificultate de implementare al programului si efortul, folosind ecuatiile descrise mai sus.

#  3.LINII DE COD LOC

 Produsul software poate fi evaluat in mod direct fie indirect. Prin evaluarea directa a procesului de inginerie software se intelege determinarea costurilor si a eforturilor asociate. Ea presupune calculul numarului liniilor de cod (LOC - lines of code) scrise, determinarea vitezei de executie, a dimensiunii memoriei, precum si a numarului de defecte raportat intr-un anumit interval de timp.

 Evaluarea indirecta a produsului reprezinta in fapt o analiza a functionalitatii, calitatii, complexitatii, eficientei, fiabilitatii, intretinerii si multor altor caracteristici.

 Costul si efortul necesar pentru a dezvolta software, calculul numarului de linii de cod (LOC) precum si alte estimari directe sunt relativ usor de estimat initial. Totusi, calitatea si functionalitatea sau eficienta si intretinerea sunt mult mai dificil de evaluat si pot fi masurate doar in mod indirect.

 Metodele de evaluare ale produsului pot fi descrise dupa cum urmeaza:

* Evaluarea productiva se concentreaza asupra rezultatelor finale ale procesului de inginerie software;
* Evaluarea calitativa ofera o indicatie a cat de aproape este produsul software de cerintele implicite si explicite ale clientului;
* Evaluarea tehnica evidentiaza mai degraba caracteristicile produsului software (ex.: complexitatea logica, gradul de modularizare) decat procesul prin care acesta a fost dezvoltat;
* Evaluarea dimensionala este utilizata pentru a "colecta" evaluarile directe ale rezultatelor si calitatii procesului de inginerie software;
* Evaluarea functionala ofera o evaluare indirecta;
* Evaluarea orientata pe resursele umane ofera informatii asupra modului in care programatorii dezvolta un produs software precum si asupra perceptiei eficientei instrumentelor si modelelor de dezvoltare.

 In continuare vom face o abordare detaliata a doua dintre metodele de evaluare expuse ce sunt mai frecvent utilizate de catre casele de soft.

1. Metoda evaluarii dimensionale

 Evaluarea dimensionala a produsului software reprezinta o estimare directa a acestuia precum si a procesului prin care el este dezvoltat. Daca un manager de proiect mentine inregistrari simple, poate fi creat un tabel cu datele ordonate dupa criteriul dimensiunii. Pentru fiecare proiect, datele dimensionale uzuale sunt:

* efortul estimeaza necesarul de resurse umane si se masoara in programatori-pe-luna sau programatori-pe-an;
* KLOC (Kilo Lines of Code) - mii de linii de cod;
* valoarea este exprimarea banesca a efortului;
* pagini de documentatie;
* numarul de erori raportate de utilizatori intr-o perioada de timp (de pilda un an).
* numarul de programatori care au lucrat la dezvoltarea produsului software.

Din datele primare continute intr-un astfel de tabel poate fi realizata o evaluare a productivitatii si una a calitatii, orientate dimensional, pentru fiecare proiect in parte:

Productivitatea = KLOC / Programatori-pe-luna

Calitatea = Numar de erori / KLOC

In completare, pot fi calculati alti parametrii interesanti:

Cost = Valoare / KLOC

Documentatie = Pagini de documentatie / KLOC

 Utilizarea parametrilor dimensionali (KLOC, efort, etc.) este controversata si ei nu sunt universal acceptati ca reprezentand cea mai buna metoda de evaluare a procesului de dezvoltare software. Controversa se invarte in jurul utilizarii liniilor de cod LOC ca marime principala. Sustinatorii variabilei LOC afirma ca aceasta este un artefact al tuturor proiectelor de dezvoltare software si poate fi usor calculata, ca multe modele de estimare utilizeaza LOC sau KLOC ca date de intrare principale si ca exista deja o literatura imensa (plus date asociate) dedicata LOC. Pe de alta parte, opozantii reclama ca variabila LOC este dependenta de limbajul de programare, ca LOC poate penaliza programe bine proiectate dar scurte, ca nu se poate asocia usor limbajelor neprocedurale si ca utilizarea ei in estimare necesita un nivel de detaliere care poate fi dificil de obtinut (ex: managerul de proiect trebuie sa estimeze numarul de linii de cod ce trebuie produse cu mult inainte ca analiza si proiectul programului sa fi fost incheiate).

2. Evaluarea Functionala

 Parametrii ce caracterizeaza din punct de vedere functional produsul software reprezinta o evaluare indirecta a acestuia si a procesului prin care el este dezvoltat. Evitand calculul LOC, parametri functionali se concentreaza asupra "functionabilitatii" sau "utilitatii" programului. Acest tip de evaluare a fost propus pentru o abordare prin masurarea productivitatii, numita metoda scorului functional. Scorul functional (SF) este obtinut utilizand o relatie empirica bazata pe estimari calculabile ale domeniului de informatie al produsului precum si pe evaluari ale complexitatii aplicatiei.

 Valorile domeniului de informatie se definesc in urmatorul mod:

* Numarul de intrari a utilizatorului: Fiecare intrare a utilizatorului care furnizeaza aplicatiei date distincte orientate catre aceasta este luata in calcul. Intrarile vor trebui distinse de interogari, care sunt calculate separat.
* Numarul de iesiri a utilizatorului: Fiecare iesire catre utilizator, care furnizeaza acestuia informatii orientate catre aplicatie, este luata in calcul. In acest context termenul "iesire" se refera la rapoarte, ecrane, mesaje de eroare, etc. Datele individuale ale unui raport nu sunt calculate separat.
* Numarul de interogari a utilizatorului: O interogare este definita ca o intrare on-line ce are drept rezultat generarea unui raspuns imediat al aplicatiei sub forma unei iesiri on-line. Fiecare interogare distincta este luata in calcul.
* Numarul de fisiere: Fiecare fisier logic de tip "master", cum ar fi o colectie logica de date care poate fi parte a unei baze de date largi sau a unui fisier individual, este luat in calcul.
* Numarul de interfete externe: Toate interfetele citibile de catre masina (fisiere de date pe banda sau disc dur) care sunt utilizate pentru a transmite informatii catre alt sistem sunt luate in calcul.

 Odata ce datele de mai sus au fost colectate, un indice de complexitate este asociat fiecarui calcul. Organizatiile care utilizeaza metoda scorului functional dezvolta criterii pentru a stabili faptul daca o anumita intrare este simpla, medie sau complexa. Bineinteles, determinarea complexitatii este un proces relativ subiectiv. Pentru a calcula scorul functional, este utilizata urmatoarea relatie:

SF = totalul-de-calcul \* 0.65 + 0.01 \* SUM (Fi)

unde totalul-de-calcul este suma rezultatelor partiale obtinute prin ponderarea valorilor domeniului de informatie. Valorile constante din ecuatia de mai sus precum si factorii de influenta care sunt aplicati calculului domeniului de informatii sunt determinati empiric.

 Valorile de ajustare a complexitatii (Fi, i=1...14) se determina evaluand influenta a 14 factori:

* Necesita sistemul back-up si recovery
* Sunt necesare facilitati de comunicatii de date
* Sunt necesare functii de procesare distribuita
* Este criteriul performantei critic
* Va rula sistemul intr-un mediu operational utilizat intens
* Necesita sistemul intrari de date in regim on-line
* Necesita sistemul de intrari de date in regim on-line, ca procesul de introducere al datelor sa aiba loc pe ecrane sau prin operatiuni multiple
* Sunt fisierele actualizate on-line
* Sunt intrarile, iesirile si interogarile complexe
* Este procesul intern complex
* Este codul proiectat astfel incat sa fie reutilizat
* Sunt conversia si instalarea programului incluse in design
* Este sistemul proiectat pentru instalari multiple in organizatii diferite
* Este aplicatia proiectata astfel incat sa faciliteze modificarea si usurinta in utilizare din partea beneficiarului

 Fiecare factor este evaluat cu o nota de la 0 la 5, avand semnificatiile:

0 - Nu influenteaza;

1 - Incidental;

2 - Moderat;

3 - Mediu;

4 - Semnificativ;

5 - Esential;

 Odata ce scorul functional a fost calculat, el este utilizat intr-o maniera asemanatoare cu metoda LOC ca o masura a productivitatii, a calitatii si a altor atribute ce definesc programul:

Productivitatea = SF / Programatori-pe-luna

Calitatea = Numar Defecte / SF

Costul =Valoare / SF

Documentatia = Pagini de Documentatie / SF

 Evaluarea pe baza scorului functional a fost conceputa initial pentru a putea fi utilizata in sistemele de informatii pentru afaceri. Totusi, extinderea propusa ulterior, denumita scor caracteristic (SC), poate permite aplicarea acestei metode si in cazul programelor din domeniul sistemelor ingineresti. Scorul caracteristic este adecvat descrierii aplicatiilor in care complexitatea algoritmilor este inalta. Aplicatiile in timp real, de control al proceselor precum si cele orientate spre obiecte au tendinta de a avea o complexitate algoritmica mare si sunt prin urmare potrivite evaluarii prin metoda scorului caracteristic. Pentru a calcula acest scor valorile domeniului informational sunt din nou contorizate si ponderate. Spre deosebire de calculul scorului functional, scorul caracteristic ia in considerare inca un domeniu de informatie (algoritmi) iar valorile de ponderare sunt fixe (vezi caseta "Calculul Scorului Caracteristic"). Valoarea scorului caracteristic final se obtine din ecuatia:

SC = totalul-de-calcul \* 0.65 + 0.01 \* SUM (Fi);

 De remarcat ca scorul caracteristic ia in considerare o noua dimensiune a soft-ului, si anume algoritmii. Inversarea unei matrici, decodificarea unui sir de biti sau tratarea unei intreruperi sunt toate exemple de algoritmi. Trebuie de asemenea remarcat ca scorul caracteristic si cel functional semnifica acelasi lucru, si anume "functionalitatea" sau "utilitatea" furnizate de catre software. In fapt, amandoua evaluarile au drept rezultat aceeasi valoare a SF-ului in situatia calculului ingineresc conventional sau a aplicatiilor de tip gestiune a informatiilor. Pentru sistemele in timp real, mult mai complexe, scorul caracteristic este adesea cu 20-35% mai mare decat cel calculat prin utilizarea in exclusivitate a scorului functional. Utilizarea scorului functional - sau a celui caracteristic - este controversata.

 Partizanii ideii sustin ca SF (SC) este independent de limbajele de programare, aceasta facandu-l ideal pentru aplicatii scrise in limbaje conventionale si neprocedurale. Ei sustin de asemenea ca el este bazat pe date ce se presupun a fi cunoscute mult anterior in evolutia proiectului, facand SF (SC) mult mai atractiv ca estimare. Pe de alta parte, oponentii ideii declara ca metoda necesita putina prestidigitatie si ca realizarea calculului se face in parte pe baza unor date mai degraba subiective decat obiective; ei mai cred ca informatiile pot fi dificil de strans "dupa ce evenimentele au avut loc" si ca SF (SC) nu are o semnificatie fizica directa - el este doar un simplu numar.

3. Tehnici de Decompozitie

 Oamenii au dezvoltat o abordare naturala in rezolvarea problemelor: daca problema ce urmeaza a fi rezolvata este prea complicata, avem tendinta sa o divizam intr-o serie de sub-probleme pana cand ajungem la un nivel la care sub-problemele pot fi rezolvate. Rezolvam apoi pe rand fiecare dintre sub-probleme in speranta ca solutiile pot fi combinate pentru a forma o solutie globala. Estimarea proiectului software este o forma de rezolvare a problemei iar in majoritatea cazurilor, problema ce trebuie rezolvata (ex: dezvoltarea unei estimari de efort si cost pentru un proiect software) este mult prea complexa pentru a fi considerata ca un tot. Din acest motiv descompunem problema, o redefinim ca pe o colectie de sub-probleme cu o complexitate mai redusa si deci, speram, mai "rezolvabile".

 Anterior am definit liniile de cod si scorul functional drept date initiale de la care pornindu-se poate fi calculata productivitatea. Datele LOC si SF sunt utilizate in doua moduri pe parcursul estimarii proiectului software:

* ca variabila de estimare ce este utilizata pentru a "dimensiona" fiecare element al software-lui;
* masuratori de baza colectate de la vechile proiecte si utilizate in conjunctie cu variabilele de estimare pentru a dezvolta estimarile de efort si cost;

 Estimarile LOC si FP sunt tehnici de estimare distincte. Totusi amandoua au un numar de caracteristici comune.

 Managerul de proiect incepe prin a prezenta o descriere sintetica a functiei finale a software-lui si, pornind de la aceasta declaratie, incearca sa descompuna proiectul viitorului produs informatic in mici subfunctii care pot fi estimate fiecare individual. Variabila de estimare LOC sau SC este apoi calculata pentru fiecare sub-functie. Masuratorile de baza ale productivitatii (ex: LOC / programatori-pe-luna sau SC / programatori-pe-luna) sunt apoi aplicate celei mai potrivite variabile de estimare si este derivat efortul sau costul subfunctiei. Estimarile subfunctiei sunt combinate in scopul de a furniza o estimare globala pentru intregul proiect. Tehnicile de estimare LOC si SC difera la nivelul de detaliere necesar pentru decompozitie. Atunci cand LOC este utilizata ca variabila de estimare, decompozitia functionala este absolut esentiala si este adesea dusa la nivele considerabile de detaliere. Deoarece datele cerute pentru estimarea scorului functional sau mai mult macroscopice, nivelul de decompozitie atunci cand SC este utilizat ca variabila de estimare este considerabil mai putin detaliat. Ar trebui de asemenea luat in considerare faptul ca LOC este estimat direct in timp ce SC este determinat indirect prin estimarea numarului de intrari, iesiri, fisiere cu date, interogari si interfete externe cat si a celor 14 valori de ajustare a complexitatii descrise anterior.

 Independent de variabila de estimare care este utilizata, managerul de proiect ofera in mod uzual un domeniu de valori pentru fiecare functie descompusa in subfunctii. Utilizand date istorice sau intuitia (atunci cand orice alta metoda da gres), managerul de proiect estimeaza o valoare LOC sau SC pentru fiecare functie, in cazul cel mai optimist, cel mai probabil si cel mai pesimist. O indicatie implicita a gradului de incertitudine este furnizata atunci cand este specificata o gama de valori.

 Se calculeaza apoi o valoare asteptata pentru LOC si SC . Valoarea asteptata pentru variabila de estimare E poate fi calculata printr-o mediere a estimarilor LOC sau SF in cazurile optimist (a), probabil (m) si pesimist (b).

 De exemplu, estimarea E= (a + 4m + b) / 6 da cea mai mare credibilitate estimarii celei mai probabile (m) si urmeaza o distributie de tip beta a probabilitatii. Presupunem ca exista o foarte mica probabilitate ca LOC curent sau rezultatele SF sa se afle in afara intervalului definit de valorile estimate in cazul optimist sau pesimist.

 Utilizand tehnici statistice standard putem calcula estimarile. Totusi, va trebui notat ca o abatere bazata pe date incerte (estimate) trebuie utilizata judicios. Odata ce valoarea asteptata E pentru variabila de estimare a fost determinata, sunt utilizate productivitatiile LOC si SC. La acest moment, managerul de proiect poate aplica una sau doua abordari diferite:

* 1. Valoarea variabilei de estimare totala pentru toate subfunctiile poate fi multiplicata cu productivitatea medie corespunzatoare acelei variabile de estimare. De exemplu, daca presupunem ca este estimat un scor functional total de 310 SF si ca productivitatea este calculata pe baza relatiei SF / programatori-pe-luna avand valoarea 5.5, atunci efortul total al proiectului este:

Efort = 310 / 5.5 = 56 programatori-pe-luna

* 1. Valoarea variabilei de estimare pentru fiecare subfunctie poate fi multiplicata printr-o valoare ajustata a productivitatii bazata pe nivelul de complexitate estimat pentru fiecare subfunctie.

##

 4.Exemplu

 Sa consideram un produs software care poate fi descompus in felul urmator: Interfata utilizator, Analiza 2-D, Analiza 3-D, Organizarea structurii de date, Afisarea graficii computerizate, Controlul perifericelor, Analiza proiectarii.

 In tabelul "Estimarea costului si a efortului", coloanele 1, 2 si 3 sunt completate de managerul de proiect. Coloana 4 ("Valoarea asteptata a LOC") este calculata dupa formula E = (a + 4m + b) / 6.

 Coloanele 5 (Lei/LOC) si 6 (LOC/programator-pe-luna) exprima productivitatea si sunt derivate din date istorice. In acest caz, managerul de proiect utilizeaza valori diferite ale productivitatii pentru fiecare functie, bazate pe gradul de complexitate.

 In fine, pe baza datelor din tabel se pot calcula rubricile privind costul (coloana 7) si efortul (coloana 8), dupa formulele:

Cost = LOC asteptat \* Lei / LOC

Luni = LOC asteptat \* LOC / Programatori-pe-luna

Costul estimat al proiectului este de 6.570.000 lei iar efortul estimat este de 145 programatori-pe-luna.

 **4.BIBLIOGRAFIE**

**Zuse, H., Bollmann, P.** - *Software metrics : using measurement theory the describe the properties and scales of static complexity metrics*

 SIGPLAN Notices, vol.24, 1989

**Whale, Geoff** - *Software Metrics and Plagiarism Detection*

 Journal of Systems Software nr.13, 1990

***IEEE Standard for a Software Quality Metrics Methodology***

 IEEE Std 1061/1992

 IEEE Standards Collection Software Engineering

 IEEE Press, New York, 1994

**Scheneidewing, N.F.** - *Methodology for validating software metrics*

 IEEE Transaction on Software Engineering, vol.18, nr.5, 1992

[1],[2] **Metrici Software – Ivan Ivan si Mihai Popescu-**

http://www.software-metrics.ase.ro/articole/METRICI%20%20SOFTWARE.htm