Testarea sistemelor de calcul şi a reţelelor

CAPITOLUL 2

Caracteristici ale calității software

Fiabilitatea

- **Fiabilitatea** măsoară capacitatea unui sistem de a funcţiona fără erori într-un interval de timp dat.
- Standardul ISO/IEC 9126, defineşte fiabilitatea ca un set de atribute bazat pe capabilitatea produsului software de a-şi menţine nivelul de performanţă, în condiţii şi pe o perioadă de timp stabilite. Analiza acestei caracteristici pune accentul pe toleranţa la defecte şi corectitudine.
- Fiabilitatea este privită ca măsura încrederii pe care există în concepţia şi în capacitatea unui sistem de programe de a funcţiona corect în toate condiţiile avute în vedere de la început. Definiţia este legată de probabilitatea ca o eroare din cadrul programelor componente ale sistemului să fie activată de un set specific de date de intrare.
- Dacă o aplicaţie informatică este activată de un număr de **n** ori şi în **k** situaţii rezultatele care se obţin sunt incomplete sau incorecte, are loc întreruperea execuţiei ori sunt acceptate date de intrare cu anomalii, fiabilitatea efectivă **F** este calculată cu relaţia:

$$F = \frac{n-k}{n} = 1 - \frac{k}{n}$$

Fiabilitatea

- Fiabilitatea aplicaţiilor informatice se apreciază prin:
  - ***disponibilitate*** - implică conservarea unei capacităţi ridicate de prelucrare a unei aplicaţii software, chiar dacă apare o anomalie în textul sursă al componentelor acesteia;
  - ***integritate*** - impune ca rezultatele să fie exacte;
  - ***reproductibilitatea prelucrării*** - urmăreşte obţinerea aceloraşi rezultate, cu ajutorul unor seturi de date identice, la momente şi condiţii de execuţie diferite.

Fiabilitatea

- Factorii care influenţează în mod deosebit fiabilitatea aplicaţiilor software distribuite sunt:
  - ***funcţionalitatea*** – definită în standardul ISO/IEC 9126 ca: *un set de atribute bazate pe existenţa unui set de funcţii şi pe proprietăţile lor specificate*; funcţiile satisfac necesităţile stabilite sau implicite; datorită acestui set de atribute care definesc funcţionalitatea se demonstrează scopul căruia serveşte aplicaţia, pentru a îndeplini cerinţele; în acest timp, alte seturi arată momentul şi modul în care aplicaţia distribuită corespunde nevoilor utilizatorilor; pentru a face referire la nevoile stabilite sau implicite, se porneşte de la definiţia dată calităţii în standardul ISO 8402, ca totalitatea trăsăturilor şi caracteristicilor unui produs sau serviciu bazat pe abilitatea sa de a satisface nevoile stabilite sau implicite; atunci când o aplicaţie informatică distribuită îşi îndeplineşte scopul, nu mai este necesar să se intervină asupra sa pentru îmbunătăţirea unor funcţiuni sau pentru introducerea de noi funcţiuni; acest lucru reduce riscul apariţiei de erori, aspect care permite menţinerea fiabilităţii aplicaţiei distribuite la un nivel ridicat;

Fiabilitatea

- ***utilizabilitatea*** - este definită în ISO/IEC 9126 ca: *un set de atribute bazate pe efortul necesar pentru utilizarea produsului software şi pe evaluarea individuală a utilizării acestuia, de către un grup stabilit sau implicit de utilizatori*; în accepţiunea pe care o oferă standardul, termenul utilizator are un înţeles larg şi include aici categorii diverse de persoane ca: operatori, utilizatori finali şi indirecţi, precum şi toate categoriile care îşi defăşoară activitatea dependent, sau sub influenţa utilizării produsului software respectiv; aplicaţiile informatice distribuite sunt complexe şi oferă o gamă largă de funcţiuni; unele dintre acestea sunt uşor de înţeles şi de folosit de personal care nu necesită pregătire de specialitate; în cazul aplicaţiei informatice distribuite de contabilitate realizarea funcţiunilor de salvare sau restaurare colecţii de date sau obţinerea listei facturilor emise este la îndemâna unui operator; apar însă şi situaţii în care utilizarea unei funcţiuni necesită cunoştiinţe aprofundate precum funcţia de verificare corelaţii pentru întocmirea bilanţului contabil;

Fiabilitatea

- ***eficienţa*** – este definită în literatura de sprcialitate ca: *o corelaţie optimă între consumul de resurse implicat de utilizarea aplicaţiei informatice distribuite şi complexitatea problemei de rezolvat*; este introdusă şi noţiunea de **ieşire a produsului software**, definită ca: *o relaţie între complexitatea datelor iniţiale şi a rezultatelor finale*; în standardul ISO/IEC 9126 definiţia se referă la: *un set de atribute bazat pe relaţia dintre nivelul de performanţă al aplicaţiei informatice şi cantitatea de resurse utilizate, în anumite condiţii stabilite*; standardul ISO 8402, oferă un înţeles mai larg termenului **resurse consumate** şi include aici memoria, timpul de execuţie, precizia, serviciile de operare, mentenanţă sau suport de personal.;
- ***mentenabilitatea*** - este definită în ISO/IEC 9126 ca: *un set de atribute bazate pe efortul necesar de a face modificările specificate în produsul software respectiv*; modificările includ corecţii, îmbunătăţiri sau adaptări la modificări ale platformelor de dezvoltare sau a celor cărora le sunt destinate, modificări în cerinţe şi specificaţii funcţionale; în literatură se afirmă că, *un produs software este mentenabil în măsura în care permite o actualizare rapidă şi uşoară, astfel încât să se utilizeze în continuare, în bune condiţii*; sensul în care sunt privite aceste actualizări este mult mai restrâns, deoarece ele se referă la operaţiile realizate la nivelul sistemului de programe, al programelor şi modulelor componente, secvenţelor de instrucţiuni şi instrucţiunilor; nivelul mentenabilităţii se stabileşte pe baza datelor obţinute în etapele de proiectare, testare şi utilizare curentă;

Fiabilitatea

- ***robusteţea*** - este cunoscută şi sub numele de **toleranţă la erori**; fiabilitatea sistemelor de programe este dependentă de toleranţa la defecte hardware; facilităţi cum sunt detectarea erorilor de prelucrare şi faptul că, atunci cînd apar solicitări eronate, acestea nu sunt executate iar sistemele continuă să rămână în execuţie, conduc la creşterea robusteţei şi implicit a fiabilităţii acestora; pentru detectarea erorilor se folosesc verificări dinamice ale informaţiei şi proceduri care să semnaleze incompatibilităţi de date sau încercări de încălcare a restricţiilor de acces;
- ***portabilitatea*** – este unul dintre conceptele de baza ale programarii la nivel inalt si *asigura capacitatea reutilizarii codului existent la mutarea pe o alta platforma de lucru*; Portabilitatea software poate fi realizată la 3 nivele:
  - Programele deja instalate pot fi mutate direct pe un alt sistem de calcul (in cazul unor platforme identice sau puternic compatibilea0;
  - Programele sunt reinstalate pe un alt sistem de calcul.
  - Programele sursa pot fi recompilate pentru un alt sistem, eventual cu parametrii de compilare diferiți.

asigurarea portabilităţii sistemelor de programe contribuie la menţinerea fiabilităţii la nivel ridicat; varietatea mediilor de dezvoltare specifice fiecărui tip de calculatoare, precum şi volumul modificărilor necesar adaptării face ca în cele mai multe cazuri să se opteze pentru soluţia rescrierii acestor sisteme de programe; modificarea sistemelor existente se accepta numai dacă transferul trebuie realizat într-un timp scurt;

Fiabilitatea

- ***siguranţa în utilizare*** - stabileşte *măsura în care un sistem de programe nu permite efectuarea de modificări neautorizate sau nedorite în volumele de date, precum şi distrugerea parţială sau totală a volumelor de date*;
- ***stabilitataea*** - exprimă *rezistenţa sistemului de programe faţă de efectele generate de o modificare a datelor iniţiale, cât şi a secvenţelor de instrucţiuni care compun programele şi modulele componente ale acestuia*; în funcţie de nivelul la care se efectuează modificarea se definesc stabilitatea la nivel de instrucţiune, la nivel de secvenţă, la nivel de modul, la nivel de program şi stabilitatea globală a sistemului;
- ***testabilitatea*** - oferă utilizatorilor posibilitatea de a *evidenţia comportamentul sistemului de programe în situaţii particulare*; prin asigurarea testabilităţii se urmăreşte descoperirea cât mai multor variante de probleme rezolvabile cu aceste sisteme de programe; în practică s-a observat că un sistem de programe cu robusteţe ridicată are un nivel de fiabilitate superior şi este în acelaşi timp un sistem de programe testabil;

Fiabilitatea

- ***complexitatea*** - este caracteristica de calitate pentru care s-au dezvoltat cele mai multe sisteme de metrici şi se studiază în corelaţie cu alte caracteristici; complexitatea determină şi creşterea cheltuielilor de exploatare a aplicaţiilor informatice distribuite, precum şi eforturile necesare pentru a dezvolta noi versiuni sau pentru a rescrie aplicaţiile sau numai anumite componente;
- ***flexibilitatea*** - determină volumul de restricţii impus utilizatorilor pentru a obţine rezultate complete şi corecte prin folosirea aplicaţiilor informatice; flexibilitatea este relaţionată cu numărul opţiunilor pe care aplicaţiile informatice distribuite le pun la dispoziţia utilizatorilor.
- ***redundanţa*** – este folosită pentru a construi aplicaţii informatice fiabile; se disting două tipuri:
  - **redundanţa spaţială** - foloseşte mai multe componente decât strictul necesar pentru a implementa un sistem de programe; cu cât redundanţa este mai mare, cu atât sunt detectate sau tolerate mai multe erori;
  - **redundanţa temporală** - constă în folosirea aceluiaşi set de instrucţiuni, pentru a efectua acelaşi lucru în mod repetat, după care rezultatele sunt comparate între ele.

Fiabilitatea

- ***Timpul*** - din punct de vedere al exprimării cantitative, fiabilitatea este descrisă adesea prin intermediul unor intervale de timp. Există diferite tipuri ale unităţilor de timp;
  - **timpul calendar** - reprezintă modul de exprimare uzuală a timpului. Este util ca fiabilitatea să fie exprimată în concordanţă cu timpul calendar, deoarece oferă managerilor şi celor ce dezvoltă software, şansa de a vedea data la care sistemul atinge obiectivele referitoare la fiabilitate;
  - **timp de execuţie** – utilizat de cele mai multe modele pentru evaluraea fiabilităţii; motivul constă în faptul că aceste modele sunt superioare celor care utilizează timpul calendar. Pentru a fi capabil să exprime fiabilitatea prin intermediul timpului calendar, modelele convertesc timpul de execuţie în timp calendar într-o etapă ulterioară.

Observație! Modelele utilizate pentru calculul fiabilităţii se bazează pe defectele care apar în mod aleator. Deoarece un defect apare numai pe durata procesului de execuţie, este important ca timpul calendar să fie utilizat în calculele referitoare la fiabilitate.

Fiabilitatea

- ***Spaţiul de intrare şi profilul operaţional*** - reprezintă mulţimea de seturi şi stări de intrare, pe care aplicaţiile informatice trebuie să le trateze. Dacă ulterior se adaugă, pentru fiecare stare particulară, probabilitatea de a fi aleasă, se obţine profilul operaţional. Urmează să se decidă care este probabilitatea ce trebuie selectată pe durata testului, pentru fiecare stare de intrare. Diferitele probabilităţi sunt determinate astfel încât profilul operaţional să reflecte modul în care componentele sistemului de programe sunt executate în timpul unei operaţii normale.

Generalitatea

- ***Generalitatea*** – capacitatea sistemului de a rezolva cât mai multe situaţii posibile pentru clasa de probleme pentru care a fost creat.

IGR=NPI / NPRB

NPI – numărul de subprobleme implementat;

NPRB – numărul maxim de subprobleme definite.

IGR=$\Sigma\alpha_i C(SPROB_i)$ / C(PROB)

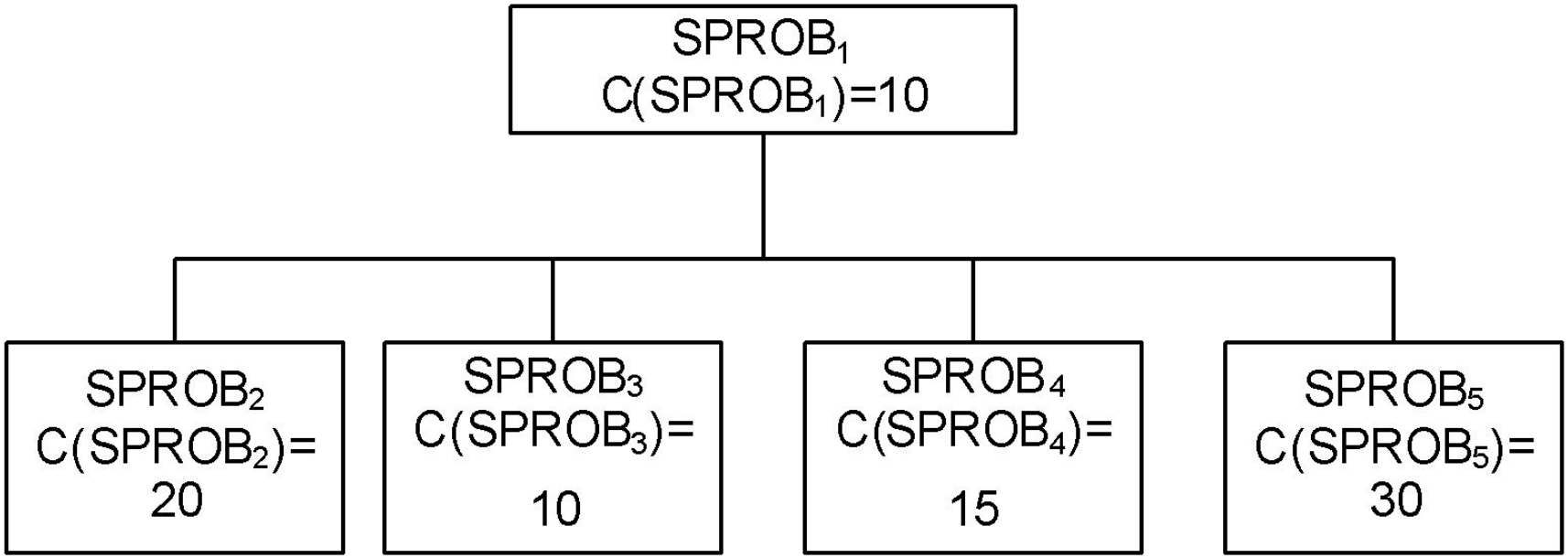
C(PROB)= $\Sigma C(SPROB_i)$ – complexitatea problemei;

$C(SPROB_i)$ – complexitatea subproblemei **i**;

α_i= 1 daca $SPROB_i$ este implementată

0 în rest

Generalitatea



Dacă se presupune că sunt implementate numai subproblemele $SPROB_1$ şi $SPROB_3$ atunci indicatorul de generalitate este:

IGR = (10 + 15) / (10 + 20 + 15 + 30) = 25 / 75 = 0,33

Rezultatul indică un nivel redus de generalitate ce corespunde soluţiei, trebuind să se treacă la ameliorarea sa.

Generalitatea -exemplu

- Se consideră un program pentru simularea funcționării unei maşini electrice asincrone. Este o problemă complexă pentru care pot fi luate in considerare trei mari cazuri, în funcție de modelul utilizat;
  - simularea fără luarea în considerare a saturației;
  - simularea cu luarea în considerare a saturației (metoda clasica);
  - simularea cu luarea în considerare a saturației (metoda înbunătățită).

Pentru fiecare din cele trei metode trebuie rezolvate următoarele subprobleme:

- introducerea parametrilor modelului;
- integrarea ecuațiilor diferențiale ale modelului;
- afişarea grafică a rezultatelor;
- efectuarea unor măsurători pe graficele obținute.

Corectitudinea

Pentru fiecare aplicaţie se elaborează specificaţii ce includ:

- descrierea datelor de intrare;
- descrierea rezultatelor;
- descrierea formulelor de calcul;
- prezentarea de comenzi;
- prezentarea restricţiilor privind sortarea datelor;
- definirea cheilor;
- prezentarea seturilor de date de test.

Corectitudinea unei aplicaţii se evidenţiază **prin testare**.

Pentru seturile de date **SDT1, SDT2, …, SDTn** *ar trebui* să obţinem rezultatele **RT1, RT2, …,RTn.**

În realitate , prin testare se vor identifica seturi **STDi** pentru care rezultatul este **RPi** în loc de **RTi**.

Corectitudinea

Latura practică a aplicaţiilor complexe legată de corectitudine are la bază următoarele ipoteze:

- existenţa a **NT** seturi de date de test;
- existenţa a **NT** rezultate considerate corecte şi incluse în specificaţii;
- asocierea unei structuri arborescente aplicaţiei informatice;
- punerea în corespondenţă a celor NT rezultate RT_i cu cele NMT module terminale din structura arborescentă:
  - în cazul în care numărul modulelor terminale este egal cu **NT**, **NMT = NT** procesul de testare este considerat complet;
  - în cazul în care **NMT > NT** rezultă că testarea este incompletă, **NMT – NT** este numărul de module finale care nu intră în teste şi odată cu ele nu intră în teste fluxurile parţiale sau totale de la modulul apelator până la ele;
  - în cazul în care **NMT < NT** există module finale supuse testărilor repetate, caz explicabil atunci când aceste module prezintă o importanţă specială în economia aplicaţiei.

Corectitudinea

Procesul de testare pentru corectitudine este definit riguros prin:

- numărul de testări;
- durata de timp limitată a procesului de testare;
- rapoarte detaliate privind rezultatul testării pentru reluarea etapelor din ciclul de dezvoltare în vederea efectuării de corecţii.

ICP $\in [0,1]$ indicator de corectitudine:

ICP=0 toate testele au condus la rezultate diferite de cele aşteptate;

ICP=1 toate testele au condus la rezultatele estimate (nu au fost puse in evidență erori);

0<ICP<1 necesită introducerea unei proceduri de estimare.

- în fiecare modul apar erori pe diverse niveluri de agregare; eroarea datorată setului **STj** de tipul **h** se va nota **ERRhj**;
- pentru fiecare tip de eroare se stabileşte un coeficient de importanță $\mathbf{PC_1 \ldots PC_{NE}}$ (**NE**- numărul de erori).

Corectitudinea

$$ICP = \begin{cases} 0, \text{ daca } \sum_{j=1}^{NT}\sum_{h=1}^{NE} ERR_{hj} = 0 \\ \dfrac{\sum_{j=1}^{NT}\sum_{h=1}^{NE} PC_h \cdot ERR_{hj}}{\sum_{j=1}^{NT}\sum_{h=1}^{NE} ERR_{hj}}, \text{ in caz contrar} \end{cases}$$

Corectitudinea

Exemplu: Vom considera o aplicație căreia i se aplică 4 seturi de date de test, $\mathbf{SDT}_j$. Pentru fiecare test se constată apariția a două tipuri de erori, **ERR1** cu PC1=0.7 şi **ERR2** cu PC2=0.3. Frecvența apariției erorilor pentru cele 4 teste este cea din tabelul de mai jos.

Set date de test	Număr erori de tip 1	Număr erori de tip 2
$\mathbf{SDT}_1$	5	3
$\mathbf{SDT}_2$	2	4
$\mathbf{SDT}_3$	7	1
$\mathbf{SDT}_4$	9	5
TOTAL	23	13

Indicatorul de corectitudine va avea în acest caz valoarea:

$$ICP = \frac{\sum_{j}^{4}\sum_{h=1}^{2} PC_h \cdot ERR_{hj}}{\sum_{j}^{4}\sum_{h=1}^{2} ERR_{hj}} = \frac{(0.7 \cdot 23 + 0.3 \cdot 13)}{23 + 13} = \frac{20}{36} = 0.55$$

Eficiența

- **Eficacitatea** măsoară avantajele pe care le generează aplicaţia analizată.
- Poate fi definită la diverse nivele:
  - la nivelul utilizatorilor aplicaţiei;
  - la nivelul administratorilor aplicaţiei;
  - la nivelul dezvoltatorilor de aplicaţii informatice.

Fie o aplicație cu **NU** utilizatori:

- $\mathbf{U_1, U_2, \dots, U_{NU}}$ – utilizatorii unei aplicații;
- $\mathbf{COST_1, COST_2, \dots, COST_{NU}}$ – costuri la utilizator pentru soluţionarea problemei în absenţa aplicaţiei;
- $\mathbf{CAPI_1, CAPI_2, \dots, CAPI_{NU}}$ costuri la utilizator pentru soluţionarea problemei cu ajutorul aplicaţiei;
- **EUAI** – indicatorul de ***eficacitate la utilizator*** al introducerii aplicației informatice **AI**:

Eficiența

$$EUAI = \frac{\sum_{j=1}^{NU} CAPI_j}{\sum_{j=1}^{NU} COST_j}$$

- Pentru ca aplicaţia să fie interesantă pentru utilizator, este necesar ca **EUAI<1**.
- Daca **EUAI≥1** utilizatorul nu numai că nu va economisi nimic, dar va avea şi pierderi.

Eficiența

Exemplu: Se considera aplicația **AI** cu 4 utilizatori $\mathbf{U_1}$,$\mathbf{U_2}$,$\mathbf{U_3}$,$\mathbf{U_4}$ și costurile din tabel:

	U_1	U_2	U_3	U_4
COST	9	8	6	7
CAPI	7	5	4	4

Rezultă:

$$EUAI = \frac{7+5+4+4}{9+8+6+7} = \frac{20}{30} = 0.66$$

Eficiența

Fie o aplicație pentru a cărei dezvoltare sunt necesare **NM** module:

- **MOD_1, MOD_2, … , MOD_{NM}** – numărul de module utilizate pentru realizarea aplicaţiei;
- **$CMOD_1$, $CMOD_2$, … ,$CMOD_{NM}$** – costurile de implementare cu tehnologii necunoscute anterior de către dezvoltator;
- **CRM_1, CRM_2, … , CRM_{NM}** - costurile de implementare cu tehnologii cunoscute şi experimentate anterior de către dezvoltator dar cu performanțe inferioare;
- **EDAI** – indicatorul de ***eficacitate la dezvoltator***:

$$EDAI = \frac{\sum_{i=1}^{NM} CMOD_i + \text{cheltuieli de instruire - performanta}}{\sum_{i=1}^{NM} CRM_i + \text{efectul negativ al neperformantei}}$$

Eficiența

Exemplu: Se consideră o aplicație realizată cu 4 module **MOD_1**, **MOD_2** , **MOD_3**, **MOD_4**. Costurile în tehnologie veche şi tehnologie nouă sunt cele din tabelul de mai jos:

		Tehnologie veche	Tehnologie nouă
Costuri	MOD_1	10	20
	MOD_2	15	20
	MOD_3	30	50
	MOD_4	40	60
Cheltuieli cu învăţarea tehnologiei		0	70
Performanţă		70	90

Rezultă:

$$EDAI = \frac{(20+20+50+60)+70-90}{(10+15+30+40)+70} = \frac{130}{165} = 0{,}78$$

Eficiența

$\mathbf{V_1}$ – variantă costisitoare pentru dezvoltator dar uşoară pentru administrator;

$\mathbf{V_2}$ – variantă uşoară pentru dezvoltator dar foarte dificilă pentru administrator.

Costurile aferente acestor variante de module sunt:

$\mathbf{CV_1 = CDC_1 + CAL_1 * NRL}$

unde:

$\mathbf{CDC_1}$ – reprezintă cheltuieli de dezvoltare a componentei de administrare pentru $\mathbf{V_1}$

$\mathbf{CAL_1}$ – reprezintă cheltuieli lunare de administrare cu $\mathbf{V_1}$

NRL – numărul de luni de administrare;

$\mathbf{CV_2 = CDC_2 + CAL_2 * NRL}$

unde:

$\mathbf{CDC_2}$ – reprezintă cheltuieli de dezvoltare a componentei de administrare pentru $\mathbf{V_2}$

$\mathbf{CAL_2}$ – reprezintă cheltuieli lunare de administrare cu $\mathbf{V_2}$

Atunci, eficacitatea la administrator **EAAI** se calculează ca fiind:

Eficiența

$$EAAI = \frac{CV1}{CV2}$$

- **Exemplu:** Dacă se consideră pentru varianta $\mathbf{V_1}$ cheltuielile de dezvoltare a componentei administrator ca fiind $\mathbf{CDC_1} = 100$ şi cheltuielile de administrare a componentei dezvoltate ca fiind $\mathbf{CAL_1} = 20$, pentru varianta $\mathbf{V_2}$ cheltuielile de dezvoltare a componentei $\mathbf{CDC_2} = 50$ şi cheltuielile de administrare a componentei $\mathbf{CAL_2} = 70$, pentru 36 de luni, rezultă eficacitatea administrator

$$EAAI = \frac{100 + 20 \cdot 36}{50 + 70 \cdot 36} = \frac{820}{2570} = 0{,}32$$

Eficiența

- ***Indicatorul de eficacitate agregat*** se calculează cu relația:

EUAD=PU*EUAI +PD*EDAI+PA*EAAI

unde:

PU, **PD**, **PA** ponderi (coeficienți de importanță)

PU,**PD**,**PA**$\in$(0,1)

PU+**PD**+**PA**=1

- **Exemplu:** Dacă pentru aplicația din exemplele precedente se consideră PU=0,6, PD=0,25, PA=0,15

atunci:

EUAD=0,6*0,66 +0,25*0.78+0,15*0,32=0,64

Continuitatea

Există în uz curent o serie de aplicaţii informatice caracterizate prin:

- structuri de interfeţe comune;
- cuvinte cheie comune;
- structuri de rezultate comune;
- moduri de autentificare identice;
- niveluri de validare asemănătoare;
- modalităţi de introducere a datelor fără generare de erori;
- reducerea şirurilor de caractere cu introducere repetată prin memorare, comparare, crearea şi afişarea unei liste de şiruri posibile în vederea selecţiei;

La realizarea unei noi aplicaţii informatice, interesul dezvoltatorilor este de a elimina efortul de adaptare al utilizatorilor în procesul de interacţiune om – calculator.

Continuitatea

- Din această cauză, când se proiectează o nouă aplicaţie informatică:

  - se includ în noua aplicaţie, exact soluţiile la problemele existente deja în alte aplicaţii;
  - se preiau fără modificări de poziţie, de text, de prezentare, de design interfeţe cu care utilizatorii sunt obişnuiţi;
  - se construiesc liste de valori pentru a facilita introducerea de date corecte prin selecţie;
  - se caută omogenitatea în a introduce date; toate datele se introduc prin selecţie de valori din liste; toate datele se introduc prin selecţie de variabile radio;
  - ordinea de introducere, poziţia valorilor care trebuie introduse, ţin seama de specificul grupului ţintă care alcătuieşte mulţimea utilizatorilor.

Continuitatea

- Dacă proprietarul aplicaţiei informatice distribuite doreşte maximizarea numărului de utilizatori, el trebuie să aibă în vedere:
  - efectele pozitive, pe care le generează utilizarea aplicaţiei la nivelul persoanelor, să fie vizibile;
  - efortul de adaptare a persoanelor la exigenţele aplicaţiilor să fie minime.
- Continuitatea este o caracteristică de calitate deosebit de importantă în definirea strategiilor de dezvoltare a aplicaţiilor informatice.

Continuitatea

- Continuitatea este caracteristica cu care se gestionează efortul de adaptare.
- Cu cât o aplicaţie informatică conţine mai multe inovaţii, foloseşte un vocabular mai nou, are interfeţe cu structuri diferite de cele existente, utilizatorii trebuie să încerce în mod repetat să parcurgă etape ale procesului de selecţie de opţiuni, cu succes sau fără succes în vederea soluţionării problemelor lor.
- Dacă se consideră aplicaţiile informatice $\mathbf{AI_1}$, $\mathbf{AI_2}$, având vocabularele $\mathbf{V_1}$ şi $\mathbf{V_2}$, nivelul de continuitate **ICO**, în raport cu vocabularele utilizate este dat de relaţia:

$$ICO = \frac{LG(V_1 \cap V_2)}{LG(V_1 \cup V_2)}$$

Continuitatea

Exemplul 1: Dacă:

V1 = {FILE, EDIT, INSERT, FORMAT, TOOLS, TABLE, WINDOWS, HELP}
V2 = {FILE, EDIT, INSERT, MODIFY, INPUT, PRINT}

$V_1 \cap V_2$= {FILE, EDIT, INSERT}

$V_1 \cup V_2$= {FILE, EDIT, INSERT, FORMAT, TOOLS, TABLE, WINDOWS, HELP, MODIFY, INPUT, PRINT}

$$LG(V_1 \cap V_2) = 3$$

$$LG(V_1 \cup V_2) = 11$$

$$ICO = \frac{3}{11} = 0{,}27$$

Continuitatea

- Dacă interfeţele au toate câmpurile comune dar se schimbă poziţia câmpurilor de la interfaţa I_1 la interfaţa I_2 atunci indicatorul de concordanţă ICNC, ce priveşte poziţiile ocupate de câmpuri în cele două interfeţe se calculează după formula:

$$ICNC = \frac{\sum_{i=1}^{N} \left| poz_{1i} - poz_{2i} \right|}{N^2}$$

unde:

poz_{1i} - reprezintă poziţia câmpului **i** în interfaţa 1;
poz_{2i} - reprezintă poziţia câmpului **i** în interfaţa 2;
N - este numărul de câmpuri din interfeţe.

Continuitatea

Exemplul 2: Se consideră două interfețe în care pozițiile câmpurilor de achiziție a datelor sunt :

1	câmpul 1
2	câmpul 2
3	câmpul 3
4	câmpul 4
5	câmpul 5
6	câmpul 6
7	câmpul 7

Poziţia câmpurilor în interfaţa I_1

1	câmpul 6
2	câmpul 5
3	câmpul 2
4	câmpul 3
5	câmpul 7
6	câmpul 1
7	câmpul 4

Poziţia câmpurilor în interfaţa I_2

Continuitatea

Câmpurile din interfaţa I_1 îşi modifică poziţia în interfaţa I_2 cu următoarele poziţii:

câmpul 1 = | 1 – 6 | = 5
câmpul 2 = | 2 – 3 | = 1
câmpul 3 = | 3 – 4 | = 1
câmpul 4 = | 4 – 7 | = 3
câmpul 5 = | 5 – 2 | = 3
câmpul 6 = | 6 – 1 | = 5
câmpul 7 = | 7 – 5 | = 2

Indicatorul de concordanţă ICNC, ce priveşte poziţiile ocupate de câmpuri în cele două interfeţe, este:

ICNC = (5 + 1 + 1 + 3 + 3 + 5 + 2)/ (7 * 7) = 0,41

Continuitatea

- Dacă interfeţele au numai anumite câmpuri comune şi se schimbă poziţia câmpurilor comune de la interfaţa I_1 la interfaţa I_2 atunci indicatorul de concordanţă **ICNC**, ce priveşte poziţiile ocupate de câmpurile comune în cele două interfeţe se calculează după formula:

$$ICNC = \frac{\sum_{h=1}^{k} \left| poz_{1h} - poz_{2h} \right|}{k * \max\left\{NC_1, NC_2\right\}}$$

unde:

poz_{1k} - reprezintă poziţia câmpului k în interfaţa 1;
poz_{2k} - reprezintă poziţia câmpului k în interfaţa 2;
k - este numărul comun de câmpuri din cele două interfeţe;
NC_1 - reprezintă numărul de câmpuri din interfaţa 1;
NC_2 - reprezintă numărul de câmpuri din interfaţa 2.

Continuitatea

Se consideră capturile de ecran

1	câmpul 1
2	câmpul 2
3	câmpul 3
4	câmpul 4
5	câmpul 5
6	câmpul 6
7	câmpul 7

Poziţia câmpurilor în interfaţa I_1

1	câmpul 4
2	câmpul 8
3	câmpul 3
4	câmpul 11
5	câmpul 9
6	câmpul 1
7	câmpul 10
8	câmpul 2

Poziţia câmpurilor în interfaţa I_2

Continuitatea

- Câmpurile comune din cele două interfeţe sunt: câmpul 1, câmpul 2, câmpul 3 şi câmpul 4. Aceste câmpuri îşi modifică poziţia din interfaţa I_1 în I_2 astfel:

- câmpul 1 = | 1 – 6 | = 5
- câmpul 2 = | 2 – 8 | = 6
- câmpul 3 = | 3 – 3 | = 0
- câmpul 4 = | 4 – 1 | = 3

- Indicatorul de concordanţă ICNC, ce priveşte poziţiile ocupate de câmpurile comune în cele două interfeţe devine în acest caz:

- **ICNC = (5 + 6 + 0 + 3)/ (4 * max{7; 8}) = 16 / 32 = 0,5**

Întrebări ?