ION VĂDUVA

FIABILITATEA PROGRAMELOR

ION VĂDUVA

FIABILITATEA PROGRAMELOR

Note de curs

EDITURA UNIVERSITĂȚII DIN BUCUREŞTI
2003

Referenți științifici: Prof. dr. **Ileana Popescu**
Conf. dr. **Denis Enăchescu**

Şos. Panduri, 90-92, Bucureşti - 76235; Telefon/Fax: 410.23.84
E-mail: editura@unibuc.ro
Internet: www.editura.unibuc.ro

Descrierea CIP a Bibliotecii Naționale
VĂDUVA, ION
Fiabilitatea programelor: note de curs / Ion Văduva
– Bucureşti: Editura Universității din Bucureşti, 2003
p.; cm.
Bibliogr.
ISBN 973-575-717-6

004.42

Cuvânt introductiv

Lucrarea de faţă constă din prelegerile ţinute studenţilor de la specializarea *INFORMATICĂ* a Facultăţii de Matematică şi Informatică a Universităţii din Bucureşti începând cu anul universitar 1997/1998, in cadrul unui curs opţional, care se organizează anual sau semestrial numai în funcţie de solicitările studenţilor.

Conţinutul lucrării este orientat pe modelarea proceselor legate de realizarea produselor soft, precum şi de evaluarea calităţii acestora. Se are în vedere în mod special evaluarea performanţelor în exploatare a produselor soft, adică *fiabilitatea, disponibilitatea şi mentenabilitatea* acestora. Se discută şi aspecte economice legate de minimizarea costuriulor de producţie sau eforturilor realizatorilor de soft dar şi de minimizarea costurilor de achiziţie făcute de către utilizatorii de soft.

Modelele prezentate au un pronunţat caracter matematic-aplicativ. Producătorii sau managerii firmelor de software pot găsi aici multe idei prin care îşi pot organiza în mod eficient activitatea astfel încât să asigure o calitate corespunzătoare produselor soft.

Primul capitol tratează pe scurt noţiuni generale ale teoriei stochastice a fiabilităţii sistemelor. Aici sunt prezentate repartiţiile statistice utilizate în fiabilitate şi procesele stochastice cu care se modelează fiabilitatea sistemelor (procesele Markov, procesele de naştere şi deces şi procesele Poisson).

Capitolul al doilea se ocupă de câteva modele markoviene ale fiabilităţii sistemelor software printre care cele mai reprezentative sunt: modelul Jelinski-Moranda, modelele cu intensitatea căderilor in formă de "S",modelul lui Shick-Wolverton, modelul lui Shantikumar şi câteva modele cenzurate.

Capitolul al treilea discută câteva modele de fiabilitatea programelor bazate pe procese Poisson neomogene (PPNO). Printre acestea se remarcă modelul lui Goel-Okumoto şi modelul lui Musa pentru timp operaţional.

Modelele analizate in capitolele doi şi trei işi propun să estimeze diferiti parametri statistici sau caracteristici de fiabilitate cum sunt: numărul iniţial de erori din soft N_0, funcţia de fiabilitate condiţionată, timpul mediu cumulat dintre căderi, etc.

In capitolul al patrulea se prezintă modele legate de optimizarea fiabilităţii, a costurilor şi a activităţilor de realizare sau achiziţionare a produselor software. Sunt prezentate şi câteva modele euristice, care pot avea aplicaţii de interes practic.

Intrucât in anumite situaţii pot exista informaţii apriori asupra unor parametri sau caracteristici de fiabilitate, in capitolul al cincilea se prezintă versiuni bayesiene ale modelului Jelinski-Moranda, sau a altor modele.

Capitolul al şaselea şi ultimul, prezintă câteva modele statice. Printre acestea se evidenţiază modelele de tip "captură-recaptură" (care sunt inspirate de aplicaţii din biologie). O bună parte a acestui capitol se ocupă de metricile de complexitate ale produselor soft, în legătură cu care se pot estima o serie de caracteristici de fiabilitate sau calitate a softului. Unele din modelele de aici pot fi aplicate în multe activităţi manageriale ale companiilor de software. De aceea capitolul al şaselea trebuie citit impreună cu capitolul al patrulea, mai ales de către cititorii interesaţi de aplicaţii. Cititorii interesaţi exclusiv de aplicaţii pot omite citirea secţiunii 6.4.

Trebuie menţionat faptul că există o foarte bogată literatură privind tematica acestei lucrări. Bibliografia ataşată constă numai dintr-o parte a lucrărilor pe care autorul le-a consultat şi care pot fi găsite cu uşurinţă în biblioteci.

Intenţia autorului este de a continua dezvoltarea materialului prezentat aici într-o ediţie următoare. Până atunci, lucrarea de faţă constituie un material suficient pentru suportul cursului opţional oferit studenţilor anului al patrulea al sectiei de informatică.

Autorul

CUPRINS

1 Noţiuni introductive de teoria fiabilităţii

1.1 Erori şi căderi în funcţionarea sistemelor

In limbajul cotidian cuvântul *fiabilitate* a unui sistem este sinonim cu *siguranţa în funcţionare* a acelui sistem. Fiabilitatea desemnează deci unul sau mai mulţi indicatori de calitate care descriu buna funţionare a unui sistem [2, 3, 4, 6, 12, 15, 27]. (Sistemul poate fi o maşină automată, un computer, o întreprindere, o vieţuitoare, sau chiar un program calculator). Nefuncţionarea unui sistem poate fi determinată de mai multe cauze care pot fi aleatoare sau nu; chiar dacă cauzele nu sunt întotdeauna aleatoare, ele se pot însă *manifesta* în mod aleator determinând ca sistemul să nu îsi producă *scopul* său adică să nu funcţioneze normal. De cele mai multe ori când sistemul nu funcţionează normal spunem că el are o *cădere,* iar aşa cum am văzut, aparţia căderii este aleatoare.

La un program, de exemplu [1, 2, 16, 22, 25, 35], căderea poate fi determinată fie de existenţa unei erori, fie de neîndeplinirea de către utilizator a condiţiilor cerute de execuţia sau exploatarea programului.

Un produs software parcurge de regulă un ciclu de viaţă constând din următoarele etape: *definirea cerinţelor sau specificaţiilor, proiectarea, programarea, testarea şi operarea/întreţinerea.* Datorită complexităţii foarte mari a produselor software (un program de acest tip poate avea sute de mii sau chiar milioane de linii de text sursă!), la testare nu este posibil să se descopere toate erorile. Dacă considerăm un program ca fiind reprezentat de o schemă logică, (vezi Fig.1.1), adică de un graf orientat [10, 13, 18], în care nodurile sunt instrucţiunile (blocuri funcţionale şi blocuri predicative), săgeţile indică succesiunea de execuţie a blocurilor, iar două noduri sunt privilegiate (blocurile de START şi STOP), atunci este simplu de înţeles că execuţia unui program inseamnă parcurgerea unui drum in graf de la START la STOP. Un program complex poate avea multe blocuri *predicative* (i.e. condiţii sau ramificaţii) şi deci multe astfel de drumuri. In faza de testare nu este posibil să se parcurgă toate drumurile pe care s-ar putea găsi erori; fiecare testare va corespunde unui astfel de drum. "Drumurile" netestate pot conţine deci erori. După ce programul este dat in funcţiune, un utilizator oarecare, folosind date de intrare aleatoare, neexplorate in faza de testare, poate descoperi întâmplător erori. In cazul întrunirii condiţiilor apariţiei unei astfel de erori, programul, fie se întrterupe, fie nu dă rezultate corecte; spunem că are loc o *cădere* a programului, care se manifestă deci în mod aleator.

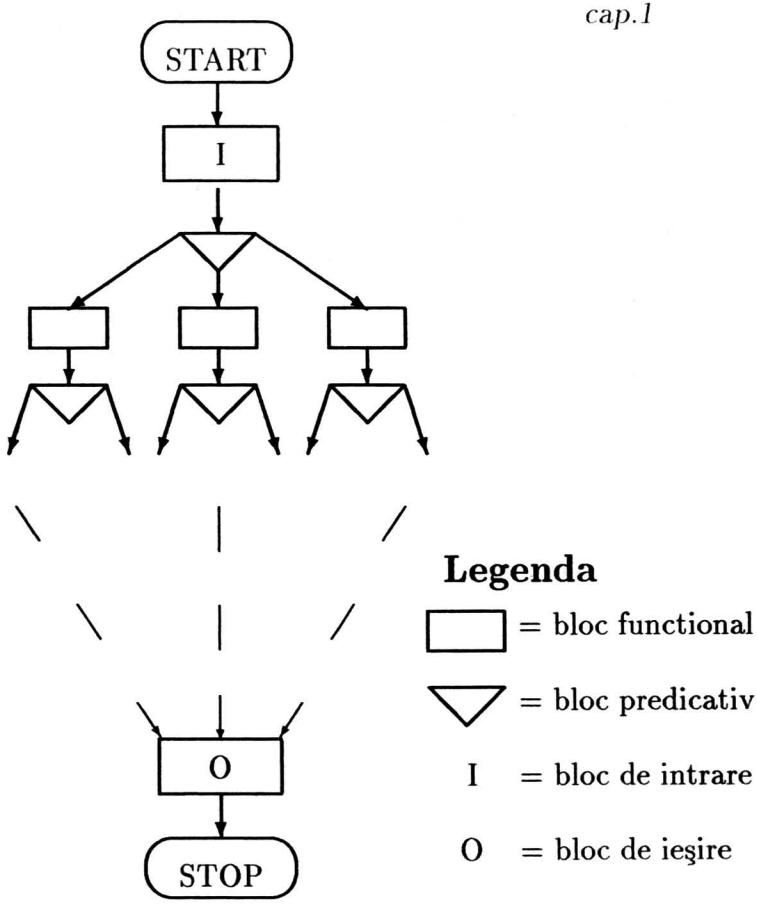


Fig.1.1 O Schemă Logică (reprezentând un program)

v_i = numărul de alternative ale blocului predicativ i, $1 \leq i \leq n$;
n = numărul de blocuri predicative ;
N_c = numărul de căi de testare $= \prod_{i=1}^{n} v_i$
$N = 2^n$ dacă $v_i = 2$.

Căderea apare astfel numai ca efect al apariţiei uneia sau mai multor erori pe *o cale de testare.*

Durata X de funcţionare fără căderi este deci o variabilă aleatoare [2, 3, 4, 6, 12, 16, 23, 25,35]; vom nota cu F funcţia de repartiţie a lui X, adică $F(t) = P(X < t)$.

Fiabilitatea unui program sau a unui sistem este probabilitatea ca acel program sau sistem să funcţioneze fără căderi, în anumite condiţii de operare,

pe un anumit interval de timp de lungime t. Fiabilitatea se notează $R(t)$ sau $\overline{F}(t) = 1 - F(t)$ şi se mai numeşte *probabilitate de supravieţuire*.

Multe sisteme create de om (sau numai *componente* ale sistemelor), pot cădea după anumite perioade de timp de funcţionare, după care are loc *întreţinerea sau repararea* lor (mentenanţa), care de asemenea se realizează în anumite perioade de timp, sistemul intrând din nou într-o perioadă de funcţionare şi a.m.d. Un astfel de sistem se numeşte *reparabil* [2]. Un produs software (sau un sistem de producţie-fabrică) este, desigur, un sistem reparabil.

La sistemele reparabile, fiabilitatea variază cu *vârsta* sistemului. Astfel dacă sistemul a fost operat o perioadă de timp de lungime t_0, atunci o caracteristică interesantă este $R_{t_0}(t) = R(t|t_0)$ care este probabilitatea ca sistemul să funcţioneze fără căderi pe intervalul $[t_0, t]$, condiţionat de faptul că el a funcţionat şi pe intervalul $[0, t_0]$, sau a fost reparat la momentul t_0. Funcţia $R(t|t_0)$ este funcţia de fiabilitate condiţionată de supravieţuirea momentului t_0 (sau funcţia de fiabilitate condiţionată de faptul că sistemul a fost revizuit sau reparat şi repus în funcţiune la momentul t_0).

1.2 Măsuri de fiabilitate şi noţiuni conexe

Una din noţiunile importante în studiul fiabilităţii programelor şi sistemelor in general, este *funcţia fiabilitate* [3, 4, 6, 12, 23, 35] (sau *funcţia de supravieţuire*) definită mai sus.

Definiţia 1.1 Fie $X \quad (X > 0)$ *o variabilă aleatoare pozitivă (durată de viaţă),* $F(x) = P(X < x)$ *- funcţia sa de repartiţie şi* $f(x) = F'(x)$ *- densitatea sa de repartiţie (presupunând că derivata există). Atunci funcţia*

$$r(x) = \frac{f(x)}{\overline{F}(x)}, \quad \overline{F}(x) \neq 0, \quad r(0) = 0. \tag{1.1}$$

se numeşte rata căderilor [3, 4, 6, 8, 35] sau rata de hazard (când nu este vorba de fiabilitate).

Dacă funcţia rată de hazard $r(x)$ este crescătoare în x, atunci repartiţia de probabilitate a lui X se numeşte [3, 4] *IFR (Increasing Failure Rate),* iar dacă $r(x)$ este descrescătoare atunci repartiţia de probabilitate se numeşte *DFR (Decreasing Failure Rate).*

Propoziţia 1.1. Intre funcţia fiabilitate şi funcţia rată de hazard există

relaţia [3, 4]

$$\overline{F}(x) = e^{-H(x)}, \quad H(x) = \int_0^x r(u)du \tag{1.2}$$

funcţia $H(x)$ fiind rata cumulată de hazard.

Demonstraţie. Din (1.1) rezultă că

$$r(x) = \frac{\left(-\overline{F}(x)\right)'}{\overline{F}(x)}$$

iar prin integrarea ultimei relaţii obţinem relaţia (1.2).

Propoziţia 1.2. Funcţia rată de hazard satisface proprietatea

$$P(\{t \le X \le t + \Delta t\} | \{X > t\}) = r(t)\Delta t. \tag{1.3}$$

Demonstraţie. Să considerăm evenimentele

$$A = \{t \le X \le t + \Delta t\}, \quad B = \{X > t\}, \quad A \subset B.$$

Atunci primul membru din (1.3) este $P(A|B)$, în care, utilizând formula lui Lagrange obţinem

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P\{t \le X \le t + \Delta t\})}{P(\{X > t\})} = r(t)\Delta t.$$

Ultima egalitate demonstrează propoziţia.

Această propoziţie dă o interpretare interesantă ratei de hazard: dacă sistemul a supravieţuit momentului t, atunci, *pericolul* ca el să cadă în intervalul mic de timp $[t, t + \Delta t]$ este proporţională cu rata de hazard $r(t)$.

Prin cuvântul *durată de viaţă* putem înţelege orice variabilă pozitivă, deci care are proprietatea $P(0-0) = 0$. O asemenea variabilă aleatoare poate fi şi *durată de îmbătrânire sau de viaţă* sau *durata unei boli cronice până la vindecare*, precum şi *durata de activitate neîntreruptă a unui salariat în aceeaşi instituţie*. Dacă durata de viaţă se referă la fiinţe vii sau la funcţionarea unui aparat sau a unui produs industrial, atunci se poate utiliza pentru modelarea fenomenului respectiv o repartiţie de tip IFR; dimpotrivă, în cazul când X este durată neîntreruptă de activitate a unui salariat in aceeaşi unitate, atunci se foloseşte pentru modelare o repartiţie DFR.

Revenind la interpretarea menţionată mai sus, relaţia (1.3) spune de fapt că $r(t)$, în cazul fiabilităţii, reprezintă *rata periculozităţii* [2, 6, 25] de cădere,

a aparatului, sistemului sau a programului. (In cazul duratei de viaţă biologice sau în demografie $r(t)$ este *pericolul de mortalitate* sau *forţa de mortalitate*). Acest *pericol* creşte în cazul repartiţiilor IFR şi descreşte în cazul repartiţiilor DFR. Pentru fiabilitatea programelor este natural să presupunem că repartiţia duratei in funcţionare pâna la cădere este de tip DFR, întrucât pe parcursul utilizării programului erorile sunt detectate şi eliminate şi deci *pericolul* apariţiei unor noi căderi scade.

In fiabilitatea programelor vor interveni şi alte caracteristici de fiabilitate. In orice program dat în exploatare de către un producător de software, există un *număr iniţial* N_0 *de erori.* Fiecare eroare se va găsi pe o *cale de testare,* adică pe un drum de la START la STOP al schemei logice. Un produs software real, care are n blocuri predicative va avea un număr exponenţial de căi de testare (vezi Fig.1.1) . De aceea programele sunt supuse unor operaţii de *testare* prin care se urmăreşte detectarea unor erori (să spunem n) care se înlătură. Din datele de test, producătorul de soft este interesat să determine (să *estimeze*) pe N_0 calculând apoi pe această bază ce efort mai este necesar pentru a înlătura pe cele $N_0 - n$ erori rămase, sau măcar o *fracţiune* din acestea.

1.3 Repartiţii statistice utilizate in fiabilitate

1.3.1 Repartiţii continue

Presupunem că durata în funcţionare până la cădere are densitate de repartiţie. In acest caz se spune că repartiţia de probabilitate este *continuă.* Vom prezenta cele mai importante repartiţii continue care pot interveni în fiabilitate.

1. *Repartiţia exponenţială* [3, 6, 23.35] (sau *exponenţială negativă*), de parametri α şi λ (notată $Exp(\alpha, \lambda)$, cu α, λ parametri reali pozitivi) are densitatea

$$f(x) = \lambda e^{-\lambda(x-\alpha)} I_{(\alpha,\infty)}(x) \tag{1.4}$$

unde $I_{(\alpha,\infty)}(x)$ este funcţia indicator. Graficul lui $f(x)$ are forma de J (literă mare) scris invers şi el este situat la dreapta axei $x = \alpha$. De aceea parametrul α se numeşte parametru de *locaţie* (valorile variabilei exponenţiale X corespunzătoare se *localizează* la dreapra lui $x = \alpha$). Parametrul λ este parametrul de *scală* (prin modificarea lui putem schimba *scala* de valori ale variabilei X).

Variabila $Exp(0,1)$ se numeşte variabila *exponenţială standard.* Notând această variabilă cu Z, avem $X = \alpha + \lambda Z$. Cea mai utilizată este variabila $Exp(0,\lambda)$ (să o notăm cu Y), pentru care funcţia de repartiţie şi funcţia fiabilitate sunt respectiv

$$F(x) = (1 - e^{-\lambda x})I_{(0,\infty)}(x), \quad \overline{F}(x) = e^{-\lambda x}, x > 0. \tag{1.4'}$$

Rata căderilor este $\lambda = const.$ adică repartiţia exponenţială este şi IFR şi DFR, sau mai precis este la limita dintre clasele IFR şi DFR. Media variabilei Y este $E(Y) = \mu = 1/\lambda$, iar media variabilei X este $E(X) = \alpha + 1/\lambda$.

Repartiţia exponenţială poate fi utilizată în fiabilitate atât ca repartiţie a duratei în funcţionare, cât şi ca repartiţie a duratei de *reparaţie.*

2. ***Repartiţia Weibull***(α, λ, ν) [6, 12, 23], cu α, λ, μ parametri reali şi pozitivi, are densitatea

$$f(x) = \frac{\nu}{\lambda}\left(\frac{x-\alpha}{\lambda}\right)^{\nu-1} e^{\left(\frac{x-\alpha}{\lambda}\right)^{\nu}} I_{(\alpha,\infty)}(x). \tag{1.5}$$

Când $\nu > 1$, graficul lui $f(x)$ are forma de clopot nesimetric cu vârful situat la dreapta axei $x = \alpha$ iar când $0 < \nu < 1$ are forma literei J (litera mare) scrisă invers. De aceea parametrul ν se numeşte parametru de *formă.* Parametrul λ se numeşte parametru de *scală* (adică modificându-l putem modifica scala de valori ale variabilei X =*timp de funcţionare*), iar parametrul α se numeşte parametru de *locaţie* (adică valorile semnificative ale lui X, ca şi graficul densităţii sale, se situează la dreapta lui $x = \alpha$).

Funcţia de repartiţie şi funcţia fiabilitate Weibull sunt respectiv

$$F(x) = \begin{cases} 0, & x \leq \alpha \\ 1 - e^{-(\frac{x-\alpha}{\lambda})^{\nu}}, & x > \alpha, \end{cases} \quad \overline{F}(x) = \begin{cases} 1, & x \leq \alpha \\ e^{-(\frac{x-\alpha}{\lambda})^{\nu}}, & x > \alpha \end{cases}. \tag{1.5'}$$

Rata căderilor este în acest caz

$$r(x) = \frac{\nu}{\lambda}\left(\frac{x-\alpha}{\lambda}\right)^{\nu-1}, \quad x > \alpha \tag{1.5"}$$

adică repartiţia Weibull este de tip DFR dacă $0 < \nu < 1$ şi de tip IFR dacă $\nu > 1$. In cazul $\nu = 1$ repartiţia Weibull se reduce la repartiţia exponenţială.

Repartiţia $Weibull(0,1,\nu)$ se numeşte repartiţia Weibull *standard.* Să notăm cu Z variabila aleatoare Weibull standard. Se arată cu uşurinţă că valoarea medie a lui Z este

$$E(Z) = \frac{1}{\nu}\Gamma\left(\frac{1}{\nu}\right)$$

și deoarece $X = \alpha + Z\lambda$ rezultă că

$$E(X) = \alpha + \lambda\frac{1}{\nu}\Gamma(\frac{1}{\nu}).$$

In formulele de mai sus cu Γ se notează funcția gamma definită de relația

$$\Gamma(p) = \int_0^\infty x^{p-1}e^{-x}dx$$

și ea satisface proprietățile

$$\frac{\Gamma(p)}{a^p} = \int_0^\infty x^{p-1}e^{-ax}dx, \quad a > 0, \quad \Gamma(k+1) = k\Gamma(k), \quad k \in \mathcal{N}.$$

Repartiția Weibull poate fi utilizată în fiabilitate atât ca repartiție a duratei în funcționare, cât mai ales ca repartiție a duratei de *îmbătrânire.*

3. *Repartiția Gamma*(α, λ, ν) [3, 6, 12, 35] cu α, λ, ν parametri reali pozitivi având aceeași semnificație ca și în cazul repartiției Weibull, are densitatea de repartiție

$$f(x) = \frac{\lambda^\nu}{\Gamma(\nu)}(x-\alpha)^{\nu-1}e^{-\lambda(x-\alpha)}I_{(\alpha,\infty)}(x). \tag{1.6}$$

Notând cu X această variabilă aleatoare și cu Y variabila *standard Gamma*$(0, 1, \nu)$ se arată cu ușurință că $E(Y) = \nu$, de unde $E(X) = \alpha + \nu/\lambda$.

Rata căderilor repartiției Gamma este dificil de calculat, iar o analiză detailată arată că ea are intervale pe care este crescătoare și intervale pe care este descrescăroare.

Repartiția Gamma este utilizată atât ca repartiție a duratelor de funcționare cât și ca repartiție a duratelor de reparație.

4. *Repartiția Lomax*(θ, a) [12, 31] are densitatea de repartiție de forma

$$f(x) = \frac{a\theta}{(1+\theta x)^{a+1}}I_{(0,\infty)}(x), \qquad a, \theta > 0. \tag{1.7}$$

Se arată că dacă X este $Exp(0, \eta\lambda)$ cu η aleator, repartizat *Gamma*$(0, b, a)$ atunci X are de fapt repartiție *Lomax*(θ, a) cu $\theta = \lambda/b$.

Funcţia de repartiţie şi funcţia de fiabilitate sunt respectiv

$$F(x) = 1 - \frac{1}{(1+\theta x)^a}, \quad \overline{F}(x) = \frac{1}{(1+\theta x)^a}, \quad x > 0. \tag{1.7'}$$

Rata căderilor este

$$r(x) = \frac{a\theta}{1+\theta x} I_{(0,\infty)}(x)$$

deci repartiţia $Lomax(\theta, a)$ este DFR. Valoarea medie a variabilei X este $E(X) = 1/(\theta(a-1))$.

Repartiţia $Lomax(\theta, a)$ caracterizează durata în funcţionare a unui echipament care este operat in *condiţii de mediu* ce nu coincid (întotdeauna) cu condiţiile de mediu pentru care a fost proiectat echipamentul.

5. *Repartiţia valorii extreme (Gumbel*(λ)*)* [3, 10] are densitatea de repartiţie

$$f(x) = \frac{1}{\lambda} exp\left\{-\frac{e^x - 1}{\lambda} + x\right\} I_{(0,\infty)}(x)\lambda > 0, \tag{1.8}$$

iar rata căderilor este

$$r(x) = \frac{e^x}{\lambda} \tag{1.8'}$$

deci această repartiţie este de tip IFR. Repartiţia $Gumbel(\lambda)$ este specifică timpului de funcţionare al unui sistem ce se deteriorează repede în timp.

6. *Repartiţia normală* $N(\mu, \sigma)$ [2, 3, 12, 34] are densitatea

$$f(x) = \frac{1}{\sigma\sqrt{(2\pi)}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad x \in R \tag{1.9}$$

unde $\mu \in R, \sigma \in R^+$ sunt parametri daţi. Graficul lui f are forma unui *clopot* simetric faţă de axa $x = \mu$ care are punctele de inflexiune $x_1 = \mu - \sigma, x_2 = \mu + \sigma$. Dacă notăm cu X variabila $N(\mu, \sigma)$, se arată că $E(X) = \mu, Var(X) = \sigma^2$. Repartiţia $N(\mu, \sigma)$ poate fi utilizată ca repartiţie a timpului de funcţionare numai dacă $\mu > 0, \mu >> 3.\sigma$, deoarece în acest caz $P(X \leq 0) \approx 0$.

Dacă notăm cu Z variabila aleatoare $N(0,1)$ atunci funcţia sa de repartiţie este

$$\Phi(x) = \frac{1}{\sqrt{(2\pi)}} \int_{-\infty}^{x} e^{-\frac{u^2}{2}} du. \tag{1.9'}$$

De aici se deduce că $P(X < x) = \Phi((x - \mu)/\sigma)$, care are deci o formă complicată dar care se poate calcula cu uşurinţă numeric, folosind formule clasice de cuadratură pentru *funcţia lui Laplace*

$$L(x) = \frac{1}{\sqrt{(2\pi)}} \int_0^x e^{-\frac{u^2}{2}} du, x > 0 \qquad (1.9'')$$

şi ţinând seama de relaţia

$$\Phi(x) = 0.5 + sgn(x)L(|x|), x \in R.$$

Rata căderilor în acest caz are o formă ce nu se poate explicita uşor şi se poate calcula tot numeric.

7. ***Repartiţia lognormală*** $LN(\mu, \sigma)$ [12, 34], este repartiţia unei variabile $X > 0$ pentru care $Y = ln(X)$ are repartiţie normală $N(\mu, \sigma)$. Deci această repartiţie este legată de cea normală pentru care este cunoscută semnificaţia parametrilor. Dacă notăm cu $m = E(X), s^2 = Var(X)$, atunci se poate deduce următoarea relaţie între parametri μ, σ şi m, s

$$\mu = ln(m) - \frac{1}{2} ln \left[\frac{s^2}{m^2} + 1\right], \quad \sigma^2 = ln[\frac{s^2}{m^2} + 1]. \qquad (1.10)$$

Funcţiile de repartiţie, de fiabilitate şi rată a căderilor se deduc în acest caz din cele ale repartiţiei normale, ele având o formă complicată, dar pot fi cu uşurinţă calculate numeric.

Repartiţia lognormală este mai degrabă utilizată în fiabilitate decât repartiţia normală. Ea este specifică duratei de reparaţie dar poate fi utilizată si pentru duratele în funcţionare.

1.3.2 Repartiţii discrete

Deşi se utilizează mai puţin ca durate de funcţionare sau reparaţie, vom prezenta aici câteva repartiţii discrete specifice teoriei fiabilităţii.

Repartiţiile discrete sunt utilizate cu precădere pentru a caracteriza numărul aleator de căderi $N(t)$ ce se produc pe un interval de timp fixat $[0, t]$ [3, 25, 34, 35], (de obicei $t = 1$ şi se foloseşte notaţia $\nu = N(1)$). Dacă este vorba de durate, se presupune că acestea se măsoară în mărimi discrete [3], adică durata $X = 0, 1, 2, ..., n, ...$ iar funcţia de frecvenţă este $f(i) = P(X = i) = p_i$. Funcţia de repartiţie şi funcţia fiabilitate, în cazul discret, sunt respectiv

$$F(x) = P(X < x) = \sum_{i, i<x} f(i), \quad \overline{F}(x) = \sum_{i, i \geq x} f(i). \qquad (1.11)$$

Dacă variabila discretă X este durată de funcţionare sau durată de reparaţie atunci funcţia *rata căderilor* este de forma

$$r(i) = \frac{f(i)}{\sum\limits_{j,j\geq x} f(j)}, \quad i = 0,1,2,...n,... \tag{1.11'}$$

Vom prezenta pe scurt câteva repartiţii discrete utilizate în fiabilitate, restrângând expunerea numai la funcţia de frecvenţă şi eventual la precizarea valorii medii şi a dispersiei.

8. *Repartiţia geometrică* $Geom(p), 0 < p < 1$, [3, 12, 34, 35] are funcţia de frecvenţă

$$P(X = i) = f(i) = pq^i, \quad q = 1 - p, \quad i = 0,1,2,...,n,... \tag{1.12}$$

Momentele de primele două ordine sunt

$$m_1 = E(X) = \frac{q}{p}, \quad m_2 = \frac{q(1+q)}{p^2} \tag{1.12'}$$

de unde rezultă

$$Var(X) = m_2 - m_1^2 = \frac{q}{p^2}. \tag{1.12"}$$

(Pentru calculul acestor momente a se vedea tehnica folosită mai jos pentru repartiţia binomială cu exponent negativ, bazată pe utilizarea funcţiei generatoare a momentelor).

9. *Repartiţia binomială cu exponent negativ* [12,34] sau *repartiţia Pascal*(k, p) $(k \in N, 0 < p < 1)$, are funcţia de frecvenţă

$$P(X = i) = f(i) = C_{k+i-1}^i p^k q^i, \quad i = 0,1,2,...,n,..., \quad q = 1 - p. \tag{1.13}$$

Pentru a afla momentele acestei repartiţii folosim *funcţia generatoare a momentelor* definită astfel

$$m_X(t) = E(e^{tX}) = \sum_{j=0}^{\infty} e^{jt} f(j), \; t < 0$$

care în cazul nostru este

$$m_X(t) = \sum_{j=0}^{\infty} C_{j+k-1}^j e^{jt} p^k q^j = \sum_{j=0}^{\infty} C_{k+j-1}^j p^k (qe^t)^j = \left[\frac{p}{1-qe^t}\right]^k.$$

Momentul de ordinul "r" este dat de relaţia

$$m_r = (m_X(t))^{(r)}|_{t=0}$$

unde cu $f^{(r)}$ se notează derivata de ordinul r a lui f. In cazul nostru avem

$$m_1 = (m_X(t))'|_{t=0} = \frac{kq}{p}, \quad m_2 = (m_X(t))^{(2)} = \frac{kqp + k(k+1)q^2}{p^2} \qquad (1.13')$$

de unde rezultă

$$m_1 = E(X) = \frac{kq}{p}, \quad Var(X) = m_2 - m_1^2 = \frac{kq}{p^2}. \qquad (1.13")$$

In ceea ce priveşte utilizarea funcţiei generatoare a momentelor, aceasta este o alternativă la utilizarea funcţiei caracteristice definită astfel

$$\varphi(t) = E(e^{itX}), \forall t \in R,$$

care este o funcţie de variabilă complexă, pe când funcţia generatoare este o funcţie de variabilă reală. Funcţia caracteristică există întotdeauna (adică pentru orice t, pe când funcţia generatoare există numai pentru $t \leq 0$. Pentru ca momentele unei variabile aleatoare X să existe, trebuie ca cele două funcţii să fie *derivabile în origine.*

10. *Repartiţia binomială* $Binom(n,p), n \in N^+, 0 < p < 1$ [6, 12,34] are funcţia de frecvenţă

$$f(i) = P(X = i) = C_n^i p^i q^{n-i}, i = 0, 1, 2, ..., n, \quad q = 1 - p. \qquad (1.14)$$

Funcţia generatoare a momentelor este

$$m_X(t) = E(e^{tX}) = \sum_{i=0}^{n} p^i q^{n-i} e^{ti} = (q + pe^t)^n \qquad (1.14')$$

de unde rezultă că

$$m_1 = E(X) = np, \quad m_2 = E(X^2) = n(n-1)p^2 + np,$$

$$Var(X) = m_2 - m_1^2 = npq. \qquad (1.1'')$$

Repartiţia binomială se foloseşte ca repartiţie a numărului de căderi (de erori detectate) pe un anume interval de timp, în care se execută n rulări ale programului.

Observaţie. Repartiţiile prezentate la pct. 8-10 de mai sus, sunt toate înrudite, ele derivând din *experienţe cu probe Bernoulli* [28]. O probă Bernoulli este un experiment ce se face asupra unui eveniment A a cărui probabilitate $p = P(A) = const.$ Acest eveniment în cazul nostru este o *rulare* a unui program care se poate termina cu un *succes,* când programul se execută până la capăt, sau cu un *eşec,* când programul se întrerupe (ca urmare a apariţiei unei erori în program pe parcursul execuţiei). Să asociem unei probe Bernoulli o variabilă aleatoare Z, numită *variabilă Bernoulli* care ia două valori: $Z = 0$ dacă proba Bernoulli conduce la un *eşec* (adică evenimentul A *nu se produce*, deci programul se întrerupe), sau $Z = 1$ in caz contrar, când proba Bernoulli conduce la un *succes,* (adică evenimentul A se produce, programul executându-se până la capăt). Rezultă că p este probabilitatea ca proba Bernoulli să producă un succes.

Cu aceste notaţii se poate arăta uşor că:

• Variabila aleatoare discretă X care reprezintă *numărul de succese pâna la producerea unui eşec* într-un sir de probe Bernoulli independente, este o *variabilă geometrică,* $Geom(p)$ [34].

• Variabila aleatoare discretă X care reprezintă *numărul de succese până la producerea a k eşecuri* într-un şir de probe Bernoulli independente, este o *variabilă* $Pascal(k, p)$ [34].

Variabilele geometrică şi Pascal iau valori în $\mathcal{N}^+$.

• Variabila aleatoare X care reprezintă *numărul de eşecuri în n probe Bernoulli independente* este o variabilă binomială $Binom(n, p)$ [34], aşa cum am menţionat şi mai sus.

Deci variabilele de la pct. **8-10** sunt *sume de variabile Bernoulli* $Z_i, i = 1, 2, ...$ independente şi identic repartizate. Folosind această observaţie se pot deduce în alt fel valorile medii şi dispersiile pentru cele trei repartiţii de probabilitate.

11. *Repartiţia* $Poisson(\lambda), \lambda > 0$, [3, 12, 25, 34] are funcţia de frecvenţă

$$f(i) = P(X = i) = \frac{\lambda^i}{i!} e^{-\lambda}, \quad i = 0, 1, ..., n, ... \tag{1.15}$$

Se observă cu uşurinţă că funcţia generatoare este

$$m_X(t) = E(e^{tX}) = e^{\lambda(e^t - 1)} \tag{1.15'}$$

de unde rezultă

$$m_1 = (m_X(t))'|_{t=0} = \lambda, \quad m_2 = \lambda(\lambda + 1), \quad Var(X) = \lambda. \qquad (1.15")$$

Repartiţia $Poisson(\lambda)$ este repartiţia evenimentelor *rare* [34] în următorul sens: ***numărul de evenimente rare, inependente, de probabilităţi constante şi egale, ce se produc pe un interval de timp constant (de ex. egal cu 1), are o repartiţie*** $Poisson(\lambda)$. Parametrul pozitiv λ reprezintă ***numărul mediu*** de evenimente rare ce se produc pe unitatea de timp considerată. Deocamdată, înţelegem prin evenimente rare ceeace ne sugerează intuiţia. Ce înseamnă evenimente rare vom preciza mai în detaliu în secţiunea următoare când vom introduce ***procesele de naştere şi deces*** şi ***procesul Poisson.***

1.4 Procese stochastice utilizate in fiabilitate

Un proces stochastic este o familie de variabile aleatoare $\{X_t, t \in T\}, T \subseteq R$ [3, 25, 35]. Parametrul t este adesea interpretat ca fiind *timpul.* Dacă $T \in R^k$, sau dacă T este o submulţime a unui spaţiu vectorial, atunci $\{X_t, t \in T\}$ este un *câmp aleator* sau un *proces cu indici multipli.* In cele ce urmează, vom considera $T \in R$.

A cunoaşte un proces stochastic înseamna a cunoaşte funcţia de repartiţie $F_{t_1,t_2,...,t_n}(x_1, x_2, ..., x_n) = P(X_{t_1} < x_1, X_{t_2} < x_2, ..., X_{t_n} < x_n)$ pentru $\forall n \in N^+$ şi $\forall t_1 < t_2 < ... < t_n \in T$. In particular trebuie cunoscută funcţia de repartiţie a lui X_t pentru orice $t \in T$. Valorile variabilelor aleatoare $X_t, t \in T$ se mai numesc uneori *stări.* iar dacă T este cel mult numărabilă, atunci procesul se numeşte *lanţ.*

Pentru a studia un proces se precizează fie mulţimea funcţiilor de repartiţie $\{F_t(x), t \in T\}$, fie se precizează anumite axiome satisfăcute de funcţiile de repartiţie din care se pot deduce (calcula) acestea.

Vom prezenta mai întâi clasa *proceselor Markov* [3, 35] care modelează o categorie însemnată de fenomene ale lumii reale ce au comportament aleator.

1.4.1 Procese Markov

Definiţia 1.2. Un proces stochastic $\{X_t, t \geq 0\}$ *(adică* $T = [0, \infty)$*) se numeşte proces Markov, dacă*

$$P(X_t < x | X_{t_1} < x_1, ..., X_{t_n} < x_n) = P(X_t < x | X_{t_n} < x_n) \qquad (1.16)$$

pentru orice $t_1 < t_2 < ... < t_n < t$.

Cu alte cuvinte repartiţia de probabilitate a procesului în viitor, depinde numai de prezent (comportamentul viitor al procesului depinde numai de prezent). Altfel spus, dându-se starea prezentă a procesului, comportarea sa viitoare este *independentă* de istoria trecută a procesului.

Procesul Markov modelează o mare parte din fenomenele aleatoare ce se întâlnesc în practică.

Dacă mulţimea S a valorilor lui X_t este discretă, atunci procesul are o *mulţime finită de stări,* iar dacă mulţimea T este discretă atunci procesul se numeşte *lanţ Markov.*

Pentru un *lanţ Markov finit* prezintă interes *probabilităţile de trecere* $p_{ij}(s,t) = P(X_t = j | X_s = i), s < t$. Deci $p_{ij}(s,t)$ este probabilitatea de a trece din starea i la momentul s in starea j la momentul ulterior t. Dacă $\pi = \{\pi_i(s)\}, i \in S$ este vectorul ce defineşte repartiţia stărilor la momentul s, presupusă cunoscută, şi dacă sunt cunoscute probabilităţile de trecere $p_{ij}(s,t)$, atunci, conform formulei probabilităţii totale, repartiţia stărilor la momentul $t, t > s$ este

$$\pi_i(t) = \sum_{j \in S} p_{ij}(s,t)\pi_j(s). \tag{1.17}$$

Matricea probabilităţilor de *trecere într-un pas* $P(t) = ||p_{ij}(t, t+1)||$ poate depinde de timpul t. Dacă această matrice nu depinde de t atunci lanţul Markov se numeşte *omogen.* Pentru un astfel de lanţ omogen, probabilitatea de trecere $p_{ij}(s,t)$ satisface condiţia $p_{ij}(s,t) = p_{ij}(t-s)$ adică depinde numai de lungimea intervalului de timp (s,t). Probabilităţile $p_{ij}(t)$ satisfac în acest caz relaţia următoare

$$p_{ij}(s+t) = \sum_{k \in S} p_{ik}(s)p_{kj}(t).$$

De aici rezultă că dacă $P = P(1)$ este matricea probabilităţilor de trecere într-un pas, atunci matricea $P(n)$ de *trecere în n paşi* este

$$P(n) = P^n. \tag{1.17'}$$

In plus dacă notăm cu $\pi(n)$ vectorul ce reprezintă repartiţia de probabilitate a stărilor la momentul $n, n > 0$ şi cu π vectorul ce reprezintă repartiţia stărilor la momentul 0, atunci (1.17) devine

$$\pi(n) = P^n\pi. \tag{1.17"}$$

Deci pentru un lanţ Markov, este suficient să cunoaştem o *repartiţie iniţială* şi probabilităţile de trecere la momentele viitoare de timp pentru a determina repartiţia lanţului la orice moment viitor de timp. In particular, dacă lanţul este omogen, atunci cunoaşterea repartiţiei iniţiale $\pi(0)$ şi a matricii de trecere într-un pas P permite calculul repartiţiei lanţului la orice moment $n, n > 0$ de timp viitor.

Lanţul Markov evoluează deci prin *treceri* succesive în timp dintr-o stare în alta. Intervalul de timp dintre două schimbări consecutive de stări (in engleză *sojourn time*), are o *repartiţie exponenţială* de un parametru ce depinde de starea vizitată. In plus, timpii dintre tranziţii sunt independenţi între ei. Acestea sunt proprietăţi structurale importante ale lanţurilor Markov.

Procesul $\{N(t), t \geq 0\}$ unde $N(t)$ este numărul de *evenimente* (treceri) realizate de un proces Markov pe intervalul de timp $[0, t]$ se numeşte *proces Markov de numărare* [3, 15, 35].

In fiabilitate, funcţionarea unei componente sau sistem presupune existenţa a cel puţin două stări (stare de funcţionare sau stare de cădere) deci lanţul Markov poate fi utilizat aici ca instrument de modelare. Numărul de căderi pe intervalul $[0, t]$ poate fi proces Markov de numărare ca şi numărul de erori detectate într-un produs soft pe intervalul de timp $[0, t]$. Un caz particular de proces Markov utilizat în fiabilitatea programelor îl constituie procesul de *naştere şi deces* ce va fi studiat in cele ce urmează.

Definiţia 1.3. Procesul Markov cu o mulţime cel mult numărabilă de stări $\{N(t), t \geq 0\}$ *se numeşte proces de naştere şi deces [3, 25, 35] dacă satisface următoarele condiţii:*

(a) Este cu creşteri independente, *adică, oricare ar fi* $t_1 < t_1 < t_3 < t_4$ *variabilele aleatoare* $N(t_2) - N(t_1)$ *şi* $N(t_4) - N(t_3)$ *sunt independente stochastic. In plus este un proces de numărare in sensul că dacă* $t_1 < t_2$ *avem* $N(t_2) - N(t_1) = N(t_2 - t_1)$.

(b) Există constantele $\{\lambda_n, n \geq 0\}$, *numite* intensităţi de natalitate *astfel încât*

$$P(\{N(t + \Delta t) = n + 1\}/\{N(t) = n\}) = \lambda_n \Delta t + O(\Delta t) \tag{1.18}$$

(c) Există constantele $\{\mu_n, n \geq 1\}$, *numite* intensităţi de mortalitate *astfel încât*

$$P(\{N(t + \Delta t) = n - 1\}/\{N(t) = n\}) = \mu_n \Delta t + O(\Delta t) \tag{1,18'}$$

(d) Probabilitatea să se nască sau să decedeze mai mult de un individ pe intervalul mic de timp $[t, t+\Delta t]$ *este neglijabilă, adică*

$$P(\{N(t+\Delta t) = n \pm i\}/\{N(t) = n\}) = O(\Delta t), \forall i > 1. \qquad (1,18'')$$

Funcţiile $O(\Delta t)$ ($O(\Delta t)$ înseamnă *neglijabil in raport cu* Δt, cand Δt este mic), tind la 0 când Δt tinde la 0, mai repede decât Δt, adică

$$\lim_{\Delta t \to 0} \frac{O(\Delta t)}{\Delta t} = 0.$$

Să observăm că clasa $\mathcal{O}$ a funcţiilor $O(\Delta t)$ satisface condiţiile:

- $(\Delta t)^2 \in \mathcal{O}$;
- dacă $O_1 \in \mathcal{O}$, $O_2 \in \mathcal{O}$ şi f_1, f_2 sunt funcţii oarecare cu valori finite, atunci $f_1 O_1 + f_2 O_2 \in \mathcal{O}$.

Ipotezele (a) şi (b) semnifică faptul că probabilităţile ca pe intervalul de timp $[t, t+\Delta t]$ să se nască sau să decedeze exact un individ sunt proporţionale cu constantele $\lambda_n, n \geq 0$ sau $\mu_n, n \geq 1$.

Ipoteza (c) spune că naşterile şi decesele sunt *rare* in sensul că probabilităţile de a se naşte sau deceda pe $[t, t+\Delta t]$ mai mult de un individ, sunt neglijabile.

Procesul de naştere şi deces definit mai sus este *proces omogen* deoarece intensităţile $\{\lambda_n, n \geq 0\}$ şi $\{\mu_n, n \geq 1\}$ nu depind de timp. Dacă însă aceste intensităţi depind de timp, adică $\lambda_n = \lambda_n(t), n \geq 0, \mu_n = \mu_n(t), n \geq 1$, atunci *procesul este neomogen.*

Recunoaştem cu uşurinţă utilitatea procesului de naştere şi deces in modelarea fiabilităţii programelor, mai ales dacă $N(t)$ este *proces de naştere pur* (când $\mu_n = 0, n \geq 1$) şi $N(t)$ este numărul de erori detectate, sau dacă $N(t)$ este *proces de moarte pur* (când $\lambda_n, n \geq 0$) şi $N(t)$ este numărul de erori rămase în soft.

Teorema 1.1. Probabilităţile $P_n(t) = P(N(t) = n), n \geq 0$ *care definesc repartiţia de probabilitate a procesului de naştere şi deces satisfac sistemul de ecuaţii diferenţiale*

$$P_0'(t) = -\lambda_0 P_0(t) + \mu_1 P_1(t) \qquad (1.19)$$

$$P_n'(t) = -(\lambda_n + \mu_n) P_n(t) + \lambda_{n-1} P_{n-1}(t) + \mu_{n+1} P_{n+1}(t), n \geq 1 \qquad (1.19')$$

Demonstraţie. In cazul $n = 0$ avem

$$P_0(t+\Delta t) = P_0(t)(1-\lambda_0 \Delta t - O(\Delta t)) + P_1(t)(1-\lambda_1 \Delta t - O(\Delta t))(\mu_1 \Delta t + O(\Delta t))$$

$$+\sum_{i,i>1} P_{n\pm i}(t)O(\Delta t)$$

Formula precedentă se deduce astfel: există 0 *indivizi* în sistem la momentul $t + \Delta t$ dacă erau 0 la momentul t şi nu s-a născut nici unul pe intervalul $[t, t + \Delta t]$, sau era 1 individ în sistem la momentul t şi pe $[t, t + \Delta t]$ nu s-a născut nici unul şi a decedat unul, sau la momentul t erau $n \pm i$ indivizi în sistem şi au decedat (s-au născut) i.

Dezvoltând parantezele în formula precedentă şi ţinând seama de proprietăţile funcţiilor $O(\Delta t)$ obţinem

$$P_0(t + \Delta t) - P_0(t) = -\lambda_0 \Delta t P_0(t) + \mu_1 \Delta t P_1(t) + O(\Delta t)$$

Dacă în ultima relaţie împărţim cu Δt şi facem $\Delta t \to 0$ obţinem formula (1.19).

In cazul $n \geq 1$, printr-un raţionament asemănător obţinem

$$P_n(t + \Delta t) = P_n(t)(1 - \lambda_n \Delta t - O(\Delta t)(1 - \mu_n \Delta t - O(\Delta t)$$

$$+P_{n-1}(t)(\lambda_{n-1}\Delta t + O(\Delta t)(1 - \mu_{n-1}\Delta t - O(\Delta t) + \sum_{i,i>1} P_{n\pm i}(t)O(\Delta t).$$

Dezvoltând parantezele, trecând P_n în membrul stâng, împărţind cu Δt şi trecând apoi la limită se obţine (1.19').

Sistemul de ecuaţii diferenţiale (1.19) şi (1.19') are soluţie unică, conform cunoscutei teoreme de existentă a solūţiei, dacă se dă o *condiţie iniţială.* O astfel de condiţie poate fi

$$P_i(0) = 1,\ i - fixat,\ P_n(0) = 0,\ n \neq i,\ \sum_{j=0}^{\infty} P_j(0) = 1. \qquad (1.20)$$

In cazul nostru trebuie să ne asigurăm că soluţia unică a sistemului defineşte o repartiţie de probabilitate a procesului, adică

$$\sum_{j=0}^{\infty} P_j(t) = 1, \quad \forall t \in [0, \infty). \qquad (1.20')$$

O teoremă a lui Feller [34] spune că o condiţie suficientă ca soluţia unică a sistemului (1.19), (1.19') să fie repartiţie de probabilitate a procesului de naştere şi deces este ca

$$\sum_{j=0}^{\infty} \frac{\mu_{j+1}}{\lambda_j} = \infty. \qquad (1.20'')$$

Sistemul (1.19), (1.19') se rezolvă uşor în *cazul staţionar* (când $P_n(t) = const = p_n, \forall t \in [0, \infty)$. In acest caz sistemul devine

$$-\lambda_0 p_0 + \mu_1 p_1 = 0, \quad -(\lambda_n + \mu_n)p_n + \lambda_{n-1}p_{n-1} + \mu_{n+1}p_{n+1} = 0, \forall n \geq 1.$$

Dacă în relaţiile precedente facem notaţia $Z_k = -\lambda_k p_k + \mu_{k+1}p_{k+1}$, aceste relaţii devin

$$Z_0 = 0, \quad Z_{k-1} = Z_k, \forall k \geq 1.$$

De aici rezultă că soluţia sistemului (1.19), (1,19') în cazul staţionar este

$$p_n = \left[\prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}}\right] p_0, \quad p_0 = \left[1 + \sum_{n=1}^{\infty} \left(\prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}}\right)\right]^{-1}.$$

Să observăm [3, 35] că în cazul procesului de naştere şi deces neomogen ecuaţiile (1.19), (1.19') rămân valabile ca şi condiţiile de existenţă a soluţiei şi a repartiţiei procesului. Pentru existenţa soluţiilor, condiţiile (1.20'), (1.20") trebuie considerate "pentru orice $t \in [0, \infty)$".

1.4.2 Procese Poisson

Un caz particular de proces de naştere şi deces (deci şi de proces Markovian) este procesul Poisson [3, 35]; el este un proces de naştere pur în care $\lambda_n = \lambda = const$ în raport cu n, iar λ se numeşte *intensitatea procesului.* Dacă $\lambda = const$ în raport cu timpul atunci procesul se numeşte *Proces Poisson Omogen* (PPO). Dacă dimpotrivă $\lambda = \lambda(t)$ atunci procesul se numeşte *Proces Poisson NeOmogen* (PPNO).

Procesul Poisson poate fi definit şi direct, prin axiome similare celor ale procesului de naştere şi deces, şi anume:

Definiţia 1.4. Procesul stochastic discret $\{N(t), t \geq 0\}$. $N(t) \in \mathbf{N}$ *este proces Poisson dacă satisface axiomele:*

(a_1). *Este proces cu creşteri independente;* $N(t)$ *este numărul de evenimente (evoluţii sau modificări) ale valorilor sistemului pe intervalul de timp* $[0, t], t < \infty$.

(b_1). *Există o funcţie cu valori finite* $\lambda(t), t \geq 0$, *numită* intensitatea procesului, *astfel încât*

$$P([N(t + \Delta t) = n + 1]/[N(t) = n]) = \lambda(t)\Delta t + O(\Delta t) \qquad \forall n \in \mathbf{N};$$

(c_1). *Evenimentele ce corespund evoluţiei procesului sunt rare în sensul că*

$$P([N(t+\Delta t)=n\pm i]/[N(t)=i])=O(\Delta t),\forall i>1,\forall n\in\mathbf{N}.$$

Folosind o cale analogă celei parcurse cu ocazia studiului proceselor de naştere şi deces, se obţine teorema

Teorema 1.2. Probabilităţile $P_n(t)=P[N(t)=n]$, $n\in\mathbf{N}$ *satisfac sistemul de ecuaţii diferenţiale*

$$P_0'(t)=-\lambda(t)P_0(t),\quad P_n'(t)=-\lambda(t)P_n(t)+\lambda(t)P_{n-1}(t),\quad n\geq 1 \tag{1.21}$$

care are soluţie unică în condiţiile iniţiale $\{\exists i\in\mathbf{N},P_i(0)=1,P_n(0)=0,n\neq i\}$ *şi această soluţie este repartiţia Poisson*

$$P_n(t)=\frac{\Lambda(t)^n}{n!}e^{-\Lambda(t)},\quad \Lambda(t)=\int_0^t\lambda(u)du,\quad \Lambda(0)=0. \tag{1.22}$$

Demonstraţie. Formulele (1.21) se obţin din (1.19), (1.19') dacă punem în acestea din urmă $\mu_n=0$.

Pentru a demonstra formula (1.22) vom considera, fără a pierde generalitatea, $i=1$. Atunci soluţia generală a primei ecuaţii (1.21) este

$$P_0(t)=ce^{-\Lambda(t)},\qquad c=const.$$

Impunând condiţia iniţială găsim $c=1$.

Pentru $n=1$ avem ecuaţia diferenţială

$$P_1'(t)=-\lambda(t)P_1(t)+\lambda(t)e^{-\Lambda(t)}$$

a cărei soluţie generală este

$$P_1(t)=ke^{-\Lambda(t)},\quad k=k(t).$$

Aplicând metoda variaţiei constantelor obţinem

$$P_1'(t)=k'(t)e^{-\Lambda(t)}-k(t)\lambda(t)e^{-\Lambda(t)}=-\lambda(t)(k(t))e^{-\Lambda(t)}+\lambda(t)e^{-\Lambda(t)}$$

care conduce la ecuaţia

$$k'(t)=\lambda(t),\quad k(t)=\Lambda(t)+c_1,\quad c_1=const.$$

Deci forma generală a soluţiei $P_1(t)$ este

$$P_1(t) = (\Lambda(t) + c_1)e^{-\Lambda(t)}$$

căreia impunându-i condiţia iniţială $P_1(0) = 0$ obţinem $c_1 = 0$ şi deci

$$P_1(t) = \Lambda(t)e^{-\Lambda(t)} = \frac{(\Lambda(t))^1}{1!}e^{-\Lambda(t)}.$$

In continuare, procedând prin inducţie după n se obţine soluţia dată de (1.22).

Funcţia $\Lambda(t)$ se numeşte *intensitate cumulată a procesului.*

Să observăm că în cazul omogen (când $\lambda(t) = \lambda = const$), această funcţie este

$$\Lambda(t) = \lambda t, \quad P_n(t) = \frac{(\lambda t)^n}{n!}e^{-\lambda t}. \tag{1.22'}$$

Procesele Poisson joacă un rol important în fiabilitatea programelor, aşa cum se va vedea în capitolele următoare. Numărul de erori ce se detectează într-un program sau, în general într-un sistem, pe intervalul de timp $[0, t]$ este de regulă un PPO, dar poate fi adesea şi un PPNO.

1.5 Estimaţii ale parametrilor

Vom preciza pe scurt modul în care se estimează parametrii principalelor repartiţii de probabilitate utilizate în fiabilitate [4,12].

1.5.1 Metode Generale

Se presupun date selecţii asupra variabilei aleatoare τ care reprezintă durata în funcţionare până la cădere. Selecţiile pot fi ne cenzurate sau cenzurate [12,17,23]. Cea mai utilizată selecţie ne cenzurată este *selecţia bernoulliană* de volum n, definită ca fiind mulţimea variabilelor aleatoare $X_1, X_2, ..., X_n$ independente şi identic repartizate ca şi τ.

In fiabilitate se folosesc adesea *selecţii cenzurate* care pot fi de *tipul I sau de tipul II.* Dacă cele n valori de selecţie se obţin observând funcţionarea a n obiecte (componente) identice până la un moment dat C, $C > 0$, atunci dacă am nota $\tau_1, \tau_2, ..., \tau_n$ duratele de funcţionare până la cădere şi dacă ţinem seama de faptul că numai o parte din componente au căzut înainte de momentul C, rezultă că în acest caz datele de selecţie sunt

$$X_i = \begin{cases} \tau_i & daca \quad \tau_i < C \\ C & daca \quad \tau_i \geq C, \end{cases} \tag{1.23}$$

care pot fi scrise prescurtat $X_i = min\{\tau_i, C\} = \tau_i \wedge C, 1 \leq i \leq n$. Selecţia $\{X_i\}, 1 \leq i \leq n$ dată de (1.23) este o *selecţie cenzurată de tipul I.* Notăm cu D mulţimea valorilor de selecţie egale cu C, $d = CardD$. Spunem uneori că selecţia de acest tip este *cenzurată la dreapta.* Schimbând scrierea inegalităţilor în (1.23) se obţine selecţia *cenzurată la stânga*

In fiabilitatea programelor se folosesc de asemenea *selecţii cenzurate de tipul II.* Dacă se dau constantele $0 = s_0 < s_1 < ... < s_m, m < n$ şi se observă funcţionarea a n componente ce cad pe intervalul $[0, s_m]$, atunci selecţia cenzurată de tipul II constă din frecvenţele n_i = numărul componentelor ce au căzut în intervalul $[s_{i-1}, s_i), 1 \leq i \leq m$, astfel încât $n = n_1 + n_2 + ... + n_m$. Selecţia cenzurată de tipul II este deci o selecţie de *date grupate.*

In fiabilitatea programelor intervin şi *selecţii cenzurate parţial grupate* [12,23]. In acest caz unele date de selectie sunt durate în funcţionare exacte şi formează o mulţime A, $a = CardA$ iar celelalte date de selecţie sunt grupate şi formează o mulţime B, $b = CardB$, $n = a + b$. Selecţia cu date cenzurate parţial grupate constă din observaţiile exacte $X_1, X_2, ..., X_a$ şi din frecvenţele $n_1, n_2, ..., n_k, b = n_1 + ... + n_m$ ale observaţiilor grupate in intervalele $[s_{i-1}, s_i), 1 \leq i \leq m$.

Pot fi considerate şi selecţii de natură generală în care intervin d date cenzurate la dreapta, l date cenzurate la stânga şi $b = n_1 + n_2 + ... + n_m$ observaţii grupate pe intervalele $[s_{i-1}, s_i), 1 \leq i \leq m$, $n = d + b + l$.

Parametrii repartiţiilor statistice ale variabilelor aleatoare se estimează prin diverse metode utilizând selecţii asupra acelor variabile. Fie $X_1, X_2, ..., X_n$ o selecţie asupra variabilei aleatoare X care are densitatea de repartiţie $f(x, \theta)$, unde θ este parametru ce poate fi estimat; el poate fi număr sau vector $\theta = (\theta_1, \theta_2, ..., \theta_k)$.

Selecţia este bernouliană, în sensul că $X_1, X_2, ..., X_n$ este o mulţime de variabile aleatoare independente, identic repartizate ca şi X.

O *estimaţie* a lui θ este o funcţie $\overline{\theta}(X_1, X_2, ..., X_n)$ care converge către θ într-unul din modurile de convergenţă ale teoriei probabilităţilor.

In particular, dacă $E[\overline{\theta}(X_1, X_2, ..., X_n)] = \theta$, spunem că $\overline{\theta}(X_1, X_2, ..., X_n) = \overline{\theta}$ este o *estimaţie nedeplasată* pentru θ; dacă $\lim_{n \to \infty} \overline{\theta}(X_1, X_2, ..., X_n) = \theta$ atunci $\overline{\theta}$ este o estimaţie deplasată pentru θ.

Dacă $\overline{\theta}(X_1, X_2, ..., X_n)$ converge *în probabilitate* către θ, când $n \to \infty$, atunci $\overline{\theta}$ se numeşte *estimaţie consistentă* pentru θ.

O estimaţie nedeplasată şi consistentă se numeşte *estimaţie absolut corectă,*

iar o estimaţie deplasată şi consistentă se numeşte *estimaţie corectă.*

Dintre metodele de estimaţie a parametrilor, două sunt cele mai importante şi anume: *metoda momentelor* şi *metoda verosimilităţii maxime.* Vom menţiona pe scurt în ce constă fiecare din metode.

• **Metoda momentelor.** Presupunem că avem de estimat k parametri $(\theta_1, \theta_2, ..., \theta_k)$ şi că momentele $m_i = E(X^i), 1 \leq i \leq k$ există şi sunt finite şi presupunem că $m_i = m_i(\theta_1, \theta_2, ..., \theta_k)$, unde funcţiile $m_i(x_1, x_2, ..., x_k)$ sunt cunoscute. Să notăm cu $\overline{m}_i$ momentele de selecţie (empirice), adică

$$\overline{m}_i = \frac{1}{n} \sum_{j=1}^{n} X_j^i. \tag{1.24}$$

Se arată cu uşurinţă că $E(\overline{m}_i) = m_i$ şi că $\overline{m}_i \to m_i, 1 \leq i \leq k$, adică momentele de selecţie $\overline{m}_i$ sunt estimaţii nedeplasate şi consistente (numite in acest caz şi *absolut corecte*) pentru momentele teoretice m_i. Putem deci scrie

$$\overline{m}_i = m_i(\theta_1, \theta_2, ..., \theta_k). \tag{1.24'}$$

Dacă sistemul de ecuaţii (2.24') se poate rezolva tunci soluţiile sale $\overline{\theta}_i = m_i^{-1}(\overline{m}_1, \overline{m}_2, ..., \overline{m}_k)$ sunt estimaţii ale parametrilor $\theta_i, 1 \leq i \leq k$, obţinute prin metoda momentelor. Dacă funcţiile $m_i(x_1, x_2, ..., x_k), 1 \leq i \leq k$ sunt şi continue, atunci $\overline{\theta}_i$ sunt estimaţii consistente pentru θ_i.

• **Metoda verosimilităţii maxime.** In cazul selecţiei bernoulliene, prin definiţie funcţia de selecţie

$$L(X_1, X_2, ..., X_n; \theta) = \prod_{i=1}^{n} f(X_i; \theta) \tag{1.25}$$

se numeşte *funcţie de verosimilitate.* Pentru o selecţie cenzurată de tipul I funcţia de verosimilitate are o formă asemănătoare dar poate fi definită şi de forma

$$L(X_1, X_2, ..., X_{n-d}, C) = \prod_{i=1}^{n-d} f(X_i) \times [F(C)]^d. \tag{1.25'}$$

Dacă selecţia este grupată (cenzurată de tipul II) atunci funcţia de verosimilitate este

$$L(n_1, n_2, ..., n_k) = \prod_{i=1}^{m} p_i^{n_i}, \quad p_i = F(s_i) - F(s_{i-1}), 1 \leq i \leq m \tag{1.25''}$$

iar dacă selecţia este parţial grupată atunci funcţia de verosimilitate este de forma

$$L(X_1, ..., X_n) = \prod_{i=1}^{a} F(X_j) \prod_{j=1}^{m} p_j^{n_j}, \quad n_1 + ... + n_m = b \tag{1.25'''}$$

Insfârşit, pentru o selecţie cenzurată generală, cu date parţial grupate şi cu date cenzurate la stânga de C^* şi la dreapta de C, funcţia de verosimilitate este

$$L(\mathbf{S}) = [F(C)]^d [1 - F(C^*)]^l \prod_{i=1}^{m} [F(s_i) - F(s_{i-1})]$$

unde cu S se notează mulţimea valorilor de selecţie. Dacă funcţia de repartiţie F depinde de de parametrii $\theta = (\theta_1, \theta_2, ..., \theta_k)'$ atunci în formulele (1.25')-(1.25"') şi în formula precedentă densitatea f şi probabilităţile $p_j, 1 \leq j \leq m$ depind de asemenea de aceşti parametri.

Metoda verosimilităţii maxime constă în a determina estimaţiile parametrilor $\hat{\theta}_i$, $1 \leq i \leq k$, din condiţia

$$L(X_1, X_2, ..., X_n) = \max_{\theta} \tag{1.26}$$

care conduce la sistemul de ecuaţii în θ_i

$$\frac{\partial L}{\partial \theta_i} = 0, \quad 1 \leq \theta_i \leq k \tag{1.26'}$$

numite şi *ecuaţii de verosimilitate.*

Desigur, estimaţiile de verosimilitate $\hat{\theta} = (\hat{\theta}_1, ..., \hat{\theta}_k)'$ sunt funcţii de valorile de selecţie $X_1, X_2, ..., X_n$; se arată că în anumite condiţii $\hat{\theta}$ sunt estimaţii consistente pentru θ.

In continuare vom prezenta ca aplicaţii estimarea parametrilor unor repartiţii care intervin în fiabilitate. Vom utiliza pentru simplificare selecţii bernoulliene, dar estimaţiile de verosimilitate se pot construi şi cu ajutorul selecţiilor cenzurate.

1.5.2 Aplicaţii în cazul repartiţiilor continue

1. In cazul repartiţiei exponenţiale avem $m = m_1 = \frac{1}{\lambda}$ de unde rezultă estimaţia

$$\overline{\lambda} = \frac{1}{\overline{m}} = \frac{n}{\sum_{i=1}^{n} X_i} \tag{1.27}$$

2. In cazul repartiţiei $Weibull(0, \lambda, \nu)$ se poate folosi metoda momentelor astfel:

Se calculează momentele de primele 2 ordine care sunt de forma

$$m_r = \frac{1}{\lambda^r}\Gamma\left(1 + \frac{r}{\nu}\right), \; r = 1, 2.$$

Se aplică apoi formulele (1.24′) din care vor rezulta parametrii estimaţi $\overline{\lambda}$ şi $\overline{\nu}$, rezolvând sistemul următor în λ şi ν

$$\overline{m}_1^2\Gamma(1 + 2/\nu) = \overline{m}_2\Gamma^2(1 + 1/\nu), \; \lambda\overline{m}_1 = \Gamma(1 + 1/\nu). \tag{1.28}$$

3. Pentru repartiţia $Gamma(0, \lambda, \nu)$ se procedează în mod asemănător. In acest caz momentele de primele două ordine sunt:

$$m_r = \frac{\Gamma(\nu + r)}{\Gamma(\nu)\lambda^r}, \; r = 1, 2$$

iar prin rezolvarea ecuaţiilor (1.24′) se vor determina estimaţiile $\overline{\lambda}$, $\overline{\nu}$.

4. In cazul repartiţiei $Lomax(\theta, a)$

$$m_1 = \frac{1}{\theta(a-1)}, \quad Var(X) = \sigma^2 = \frac{a}{\theta^2(a-1)^2(a-2)} \tag{1.29}$$

Estimând m_1 şi σ^2 cu (1.24) şi rezolvând sistemul care rezulta din egalarea expresiilor din (1.29) cu estimaţiile $\overline{X}$, s^2, se obţin estimaţiile $\overline{\theta}$, $\overline{a}$.

5. Pentru repartiţia valorii extreme ($Gumbel(\lambda)$) vom aplica *metoda verosimilităţii maxime* după cum urmează:

Funcţia de verosimilitate este

$$L(X_1, X_2, ..., X_n) = \frac{1}{\lambda^n}\exp\left\{-\sum_{i=1}^{n}\frac{e^{X_i} - 1}{\lambda} + \sum_{i=1}^{n} X_i\right\}. \tag{1.30}$$

Logaritmând (1.30) si impunând condiţia de maxim (prin anularea derivatei de ordinul intâi) avem

$$-\frac{n}{\lambda} - \frac{n}{\lambda^2} - \frac{1}{\lambda^2}\sum_{i=1}^{n} e^{X_i} = 0 \tag{1.30′}$$

de unde rezultă estimaţia

$$\overline{\lambda} = \frac{\sum_{i=1}^{n} e^{X_i} - n}{n}. \tag{1.30″}$$

6. In cazul repartiţiei normale parametri μ şi σ^2 se estimează cu media aritmetică $\overline{X}$ şi dispersia empirică s^2 date de formulele (1.24).

7. Parametri repartiţiei lognormale $LN(\mu, \sigma)$ se estimează astfel: se estimează mai intâi cu (1.24) media şi dispersia variabilei lognormale, iar apoi din formulele (1.10) se determină estimaţiile $\overline{\mu}, \overline{\sigma}$ ale parametrilor lognormalei.

1.5.2 Aplicaţii în cazul repartiţiilor discrete

8. Pentru repartiţia geometrică folosim prima formulă (1.12') de unde rezultă estimaţia parametrului p

$$\overline{p} = \frac{1}{\overline{X} + 1}. \tag{1.31}$$

9. In cazul repartiţiei $Pascal(k, p)$ estimaţiile parametrilor k, p se realizează rezolvând sistemul următor

$$\overline{X} = \frac{k(1-p)}{p}, \quad s^2 = \frac{k(1-p)}{p^2}$$

care conduce la

$$\overline{p} = \frac{\overline{X}}{s^2}, \quad \overline{k} = \frac{\overline{X}^2}{\overline{X} - s^2}. \tag{1.32}$$

10. Parametrii repartiţiei binomiale $Binom(n, p)$ se estimează pe baza formulelor (1.14") care conduc în final la estimaţiile

$$\overline{n} = \frac{\overline{X}^2}{\overline{X} - s^2}, \quad \overline{p} = \frac{\overline{X} - s^2}{\overline{X}}. \tag{1.33}$$

11. Deoarece valoarea medie a variabilei $Poisson(\lambda)$ este $m_1 = \lambda$ rezultă că estimaţia sa este $\overline{\lambda} = \overline{X}$.

In cazul *procesului* $Poisson(\lambda)$ estimaţia este aceeaşi dacă valorile de selecţie X_i reprezintă numărul de căderi pe unitatea de timp.

Dacă se cunosc *informaţii apriori* despre anumiţi parametri atunci se pot utiliza *metode Bayessiene* [24, 35] pentru determinarea estimaţiilor acestora.

1.6 Fiabilitatea sistemelor cu mai multe componente

1.6.1 Generalităţi privind fiabilitatea sistemelor

Fiind dat un sistem $\mathcal{S}$ cu n componente $C_1, C_2, ..., C_n$ astfel încât componenta C_i are variabila de stare $X_i(t)$ ($X_i(t) = 1$ [6, 12, 27] dacă sistemul funcţionează la momentul t şi $X_i(t) = 0$ dacă sistemul nu funcţionează) se pune problema determinării funcţiei de fiabilitate $R(t)$ a sistemului definită astfel

$$R(t) = P(\Phi(\tau) = 1, \forall 0 \leq \tau \leq t) \tag{1.34}$$

unde $\Phi(t)$ este variabila de stare a sistemului (adică $\Phi(t) = 1$ dacă sistemul funcţionează, $\Phi(t) = 0$ dacă sistemul nu funcţionează la momentul t).

Se pot considera şi componente cu mai multe stări [23], dar aici ne vom ocupa numai de sisteme ale căror componente au numai două stări, ca de altfel şi sistemele de programe.

Una din problemele importante ale teoriei fiabilităţii este următoarea: dându-se un sistem $\mathcal{S}$, format din componentele $C_1, C_2, ..., C_n$ ale căror funcţii fiabilitate sunt $R_1(t), R_2(t), ..., R_n(t)$, să se găsească o funcţie $h(x_1, x_2, ..., x_n)$ astfel încât fiabilitatea $R(t)$ a sistemului $\mathcal{S}$ să se calculeze după formula

$$R(t) = h(R_1(t), R_2(t), ..., R_n(t)). \tag{1.35}$$

Funţia h se determină, aşa cum se va vedea, dacă se cunoaşte *structura de fiabilitate* sau *topologia* sistemului [3, 6, 15, 23, 27]. In funcţie de această structură se determină *funcţia de structură a sistemului* definită astfel

$$\Phi(\mathbf{X}(t)) = \Phi(X_1(t), X_2(t), ..., X_n(t)), \mathbf{X}(t) = (X_1(t), X_2(t), ..., X_n(t)). \tag{1.36}$$

In cele ce urmează vom nota $X_i = X_i(t)$. Deci

$$Z = \Phi(X_1, X_2, ..., X_n) = \begin{cases} 1, daca\ sistemul\ functioneaza \\ 0,\ altfel. \end{cases} \tag{1.37}$$

Exemplu [15, 23]. Sistemul $\mathcal{S}$ este de tipul *"k din n"* dacă el cade când cel puţin k din cele n componente ale sale cad. Dacă $C_1, C_2, ..., C_m$ sunt cele $m = C_n^k$ submulţimi de componente, de cardinalitate k, atunci se deduce că

$$\Phi(X_1, X_2, ..., X_n) = \min_{1 \leq j \leq m} \max_{i \in C_j} \{X_i\}. \tag{1.38}$$

Dacă $k = 1$ atunci sistemul are componentele conectate în *paralel* (din punct de vedere al fiabilităţii este *sistem paralel*) şi

$$\Phi(X_1, X_2, ..., X_n) = \min_{1 \le j \le n} \{X_j\} = 1 - (1 - X_1)(1 - X_2)...(1 - X_n). \quad (1.38')$$

Dacă $k = n$ atunci sistemul este *sistem serie* şi

$$\Phi(X_1, X_2, ..., X_n) = \max_{1 \le j \le n} \{X_j\} = X_1 X_2 ... X_n. \quad (1.38")$$

Definiţia 1.5. Componenta C_i a unui sistem este irelevantă dacă $\Phi(0_i, x) = \Phi(1_i, x)$ unde $(0_i, x)$ este vectorul care are pe poziţia "i" valoarea 0, iar $(1_i, x)$ este vectorul ce are pe poziţia "i" valoarea 1 [23], *celelalte componente ale lui x rămânând neschimbate. Dacă $\Phi(0_i, x) = \Phi(1_i, x)$ atunci componenta C_i se numeşte semnificativă.*

Cu alte cuvinte valoarea variabilei de stare a componentei i nu schimbă valoarea funţiei de structura Φ şi deci nici fiabilitatea sistemului indiferent dacă componenta C_i a sistemului funcţionează sau nu.

Definiţia 1.6. Importanţa structurală a componentei C_i este

$$I_\Phi(i) = \frac{1}{2^{n-1}} \sum_{(x|x_i=1)} [\Phi(1_i, x) - \Phi(0_i, x)]. \quad (1.39)$$

Se observă că pentru un sistem *"k din n"* avem

$$I_\Phi(i) = \frac{C_{n-1}^{k-1}}{2^{n-1}}. \quad (1.39')$$

In particular pentru un sisten paralel/serie avem

$$I_\Phi(i) = \left(\frac{1}{2}\right)^{n-1}. \quad (1.39")$$

Definiţia 1.7. Importanţa relativă a componentei C_i este

$$RI_\Phi(i) = \frac{nI_\Phi(i)}{\sum_{j=1}^{n} I_\Phi(j)}. \quad (1.40)$$

Definiţia 1.8. Sistemul $\mathcal{S}$ este monoton [23] dacă pentru $\forall X(t) \le Y(t)$ (ordinea fiind cea lexicografică) avem $\Phi(X(t)) \le \Phi(Y(t))$. Sistemul se numeşte coerent dacă este monoton şi nu are componente irelevante [3, 7, 15, 23, 27].

Cele mai interesante sisteme construite şi operate de om sunt sistemele coerente. Pentru un astfel de sistem fiecare din componente este relevantă in sensul că comportarea ei (funcţionarea sau căderea) are influenţă determinată asupra comportării sistemului $\mathcal{S}$.

Pentru definirea funcţiei de structură Φ a unui sistem un rol important îl joacă aşa numitele mulţimi *tăietură* sau mulţimile *traseu* ce vor fi definite î n cele ce urmează.

Definiţia 1.9. O submulţime C de componente ale sistemului format din componentele $\{1,2,...,n\}$, $(C \subset \{1,2,...,n\})$ se numeşte mulţime tăietură dacă $\forall x$ cu $x_i = 0, i \in C$ şi $x_j = 1, j \in \overline{C}$ unde $\overline{C}$ este complementara lui C in raport cu mulţimea $\{1,2,...,n\}$, avem $\Phi(x) = 0$.

Cu alte cuvinte dacă toate componentele unei mulţimi tăietură cad, atunci sistemul cade, altfel sistemul funcţionează; dacă cel puţin una din componentele unei mulţimi tăietură funcţionează, atunci sistemul funcţionează. Interesează de obicei mulţimile tăietură *minimale* [6, 23, 27] adică mulţimile care au *cel mai mic număr de elemente.*

Definiţia 1.10. Submulţimea $P \subset \{1,2,...,n\}$ cu proprietatea că $\forall x$ astfel incât $x_i = 1, i \in P$ şi $x_j = 0, j \in \overline{P}$, ($\overline{P}$= complementara lui P), avem $\Phi(x) = 1$ se numeşte mulţime traseu.

Sistemul funcţionează dacă toate componentele unei mulţimi traseu funcţionea dacă cel puţin una cade, atunci sistemul cade. Şi în acest caz interesează *mulţimile traseu minimale.*

Legăturile funcţionale dintre componentele sistemului pot fi reprezentate sub forma unui graf. Un sistem poate fi deci privit ca un graf conex (orientat sau nu) care are *intrări* în anumite noduri şi *ieşiri* în alte noduri. Nodurile corespund componentelor, deci ele se pot defecta. Submulţimea tăietură este aceea care are proprietatea că dacă nodurile sale se defectează (toate), atunci sistemul cade (nu mai este asigurată conexitatea grafului). Submulţimile traseu corespund *drumrilor de la nodurile de intrare la cele de ieşire.* Dacă toate nodurile unei submulţimi traseu funcţionează atunci sistemul funcţionează.

Teorema 1.3. Dacă notăm $T_1, T_2, ..., T_m$ mulţimile tăietură minimale şi cu $P_1, P_2, ..., P_r$ mulţimile traseu minimale, atunci funcţia de structură a sistemului este [6, 23]

$$\Phi(x) = \min_{1 \le j \le m} \max_{i \in T_j} (x_i) = \max_{1 \le j \le r} \min_{i \in P_j} (x_i). \tag{1.41}$$

Demonstraţie. Când $\max_{i \in T_j}(x_i) = 0$ înseamnă că toate componentele din T_j cad. In plus sistemul cade dacă cel puţin pentru o mulţime T_j (adică pentru cel puţin un j), toate componentele cad. De aici rezultă că

$$\Phi(x) = \min_{1 \leq j \leq m} \max_{i \in T_j}(x_i).$$

In mod asemănător, dacă $\min_{i \in P_j}(x_i) = 1$ înseamnă că toate componentele din mulţimea traseu P_j funcţionează. In plus, sistemul funcţionează dacă pentru cel puţin o mulţime traseu componentele acesteia funcţionează. In concluzie

$$\Phi(x) = \min_{1 \leq j \leq r} \max_{i \in P_j}(x_i).$$

Teorema este astfel demonstrată.

Pentru un sistem "k din n" avem $m = C_n^k$.

Dacă $k = 1$ atunci $m = n$ şi $T_j = j, 1 \leq j \leq n$ şi deci

$$\Phi(x) = \min_{1 \leq j \leq n}(x_j) = \prod_{j=1}^{n} x_j. \tag{1.42}$$

In cazul $k = n$ avem o singură mulţime traseu P care are n elemente (adică conţine toate componentele sistemului). Deci funcţia de structură este

$$\Phi(x) = \max_{1 \leq j \leq n}(x_j) = 1 - \prod_{j=1}^{n}(1 - x_i) = \coprod_{j=1}^{n} x_j. \tag{1.43}$$

(Semnul $\coprod$ se citeşte "*coprod*").

Au loc următoarele teoreme

Teorema 1.4. Funcţia de structură Φ a unui sistem coerent cu n componente satisface proprietatea [15, 27]

$$\prod_{i=1}^{n} x_i \leq \Phi(x) \leq \coprod_{i=1}^{n} x_i. \tag{1.44}$$

Teorema 1.5. Fiind daţi doi vectori $n-$dimensionali x, y cu valori 0 sau 1, să introducem notaţiile [6, 15, 23]

$$xy = (x_1y_1, x_2y_2, \ldots, x_ny_n)$$

$$x \sqcup y = (x_1 \sqcup y_1, x_2 \sqcup y_2, ..., x_n \sqcup y_n),\ x_i \sqcup y_i = 1 - (1 - x_i)(1 - y_i).$$

Atunci funcţia de structură Φ a unui sistem coerent satisface proprietăţile

$$\Phi(x \sqcup y) \geq \Phi(x) \sqcup \Phi(y),\ \Phi(xy) \leq \Phi(x)\Phi(y). \tag{1.45}$$

Funcţia de structură Φ a unui sistem, în raport cu o componentă i fixată, poate fi calculată şi folosind următoarea teoremă.

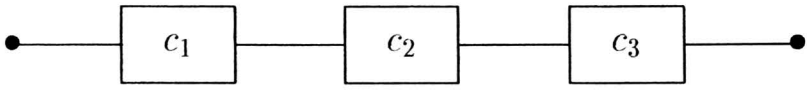
Teorema 1.6. Dacă x este vectorul $n-$dimensional al stărilor componentelor şi x_i este variabila de stare a componentei i atunci [15, 23]

$$\Phi(x) = x_i\phi(1_i, x) + (1 - x_i)\Phi(0_i, x). \tag{1.46}$$

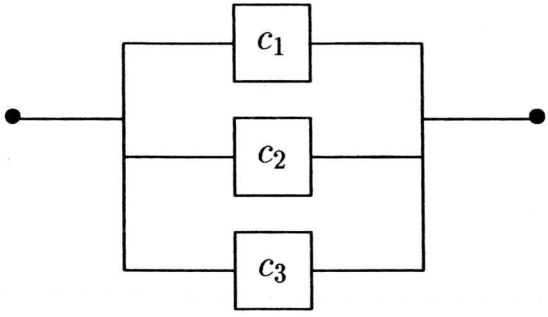
Formula (1.46) se numeşte *forma pivotală* a funcţiei de structură.

Funcţia de structură Φ se poate determina uşor folosind aşa zisa *diagramă de fiabilitate* a sistemului. Această diagramă [3, 6, 15, 23, 27] este o schemă grafică în care, prin dreptunghiuri sunt reprezentate componentele sistemului şi prin linii sunt reprezentate legăturile funcţionale ale sistemului. Dacă de ex. este vorba de o fabrică, componentele reprezintă puncte de lucru sau utilaje iar liniile indică succesiunea operaţiilor la aceste componente. Formal, diagrama de fiabilitate se rezumă la un graf conex, aşa cum am spus mai sus. Din această diagramă putem determina mulţimile tăietură şi/sau mulţimile traseu, cu care apoi determinăm funcţia de structură Φ cu formula (1.41).

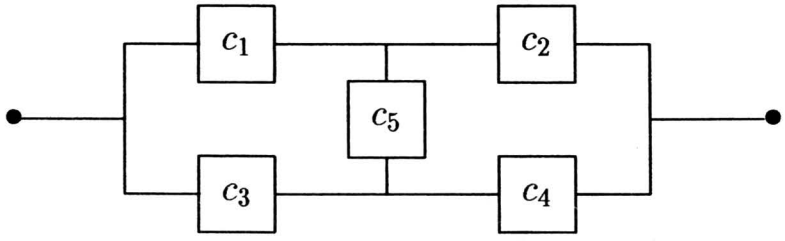
In Fig.1.2 sunt prezentate câteva diagrame de fiabilitate particulare, corespunzând sistemelor cu componente conectate în serie, în paralel şi în punte.



a.Componente conectate î n serie.



b.Componente conectate în paralel.



c. Componente conectate în punte.

Fig. 1.2 Diagrame de fiabilitate particulare.

Pentru diagramele din Fig.1.2 formulele de structură Φ sunt respectiv [3, 6, 27]

$$\Phi(x_1, x_2, x_3) = x_1 x_2 x_3; \tag{1.47a}$$

$$\Phi(x_1, x_2, x_3) = 1 - \prod_{i=1}^{3}(1 - x_i); \tag{1.47b}$$

$$\Phi(x_1, x_2, x_3, x_4, x_5) =$$

$$x_5[1 - (x_1 x_2)(1 - x_3 x_4)] + (1 - x_5)[1 - (1 - x_1)(1 - x_3)][1 - (1 - x_2)(1 - x_4)] \tag{1.47c}$$

ultima formulă putând fi dedusă şi din forma pivotală (1.46).

Funcţia de structură calculată conform regulilor menţionate se poate *simplifica* pentru a uşura efectuarea calculelor. De exemplu, funcţia de structură

a unui sistem cu trei componente paralele (adică conectate în paralel), se poate scrie [3, 6 , 15, 27]

$$\Phi(x_1, x_2, x_3) = x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3 + x_1x_2x_3$$

care este forma simplificată a formulei (1.47b) de mai sus a lui Φ.

Definiţia 1.11. Un sistem serie-paralel (notat SP) este un sistem ale cărui componente sunt conectate în serie sau în paralel sau se poate descompune în subsisteme serie-paralel.

In Fig.1.3 se prezintă un sistem SP. El se descompune în subsistemele $S_1(A, B, C), S_2(D, E)$ conectate în paralel şi în subsistemul $S_3(F, G)$.

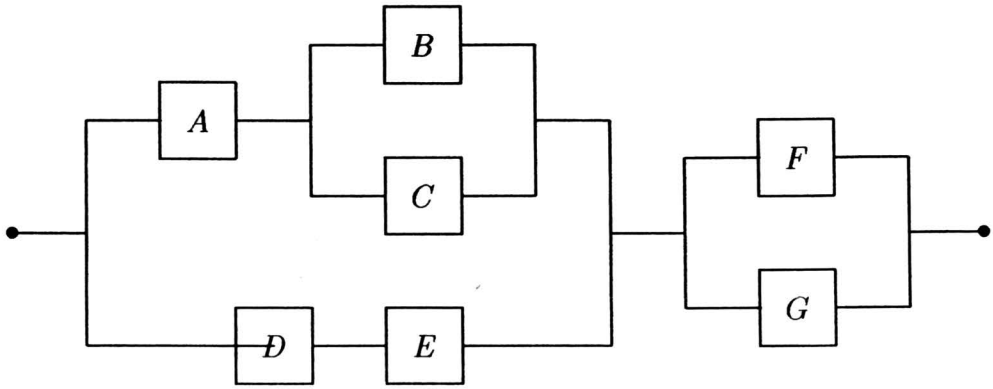


Fig. 1.3 Exemplu de sistem SP

Dacă notăm $S^* = S^*(S_1, S_2)$ subsistemul format din subsistemele S_1 şi S_2 atunci sistemul $S = S(S^*, S_3)$ din Fig.1.3 constă din subsistemele S^* şi S_3 conectate in serie.

Definiţia 1.12. Un sistem este SP-generalizat dacă fiecare componentă a sa este semnificativă şi el poate fi transformat într-un sistem SP echivalent în care unele componente se pot eventual repeta.

Multe sisteme reale, printre care şi programele modulare complexe sunt sisteme SP-generalizate.

Teorema 1.7. Funcţia de structură a unui sistem SP-generalizat satisface proprietatea

$$\min_{1\le i\le n}(x_i) \le \Phi(x) \le \max_{1\le i\le n}(x_i) \qquad (1.44')$$

unde $x = (x_1, x_2, ..., x_n)'$ *este vectorul stărilor componentelor.*

O alternativă la diagrama de structură pentru descrierea şi calculul fiabilităţii unui sistem este *arborele de avarie* [6, 15, 25, 27]. Construcţia lui se face luând în seamă *cauzele* sau condiţiile care pot produce o *cădere,* dar nu vom insista aici asupra modului de construcţie a acestuia.

Pentru un sistem dat se poate introduce funcţia sa de fiabilitate determinată direct dacă considerăm variabila aleatoare X ce reprezintă durata în funcţionare a sistemului, fără a ţine cont în mod explicit de căderile componentelor. Desigur, sistemul cade atunci când cad una sau mai multe componente importante din *structura* sa, dar noi ignorăm aceste detalii şi considerăm sistemul ca şi cum ar fi format dintr-o singură componentă. Vom avea deci de-a face cu funcţia de fiabilitate a sistemului $R(t) = P(X > t) = \overline{F}(t) = 1 - F(t)$, cu rata căderilor $h(t)$ definită în mod obişnuit şi *rata cumulată a căderilor* sistemului de forma $H(t) = \int_0^t h(u)du$.

Pentru un sistem complex, funcţia $h(t)$ are forma unei *căzi de baie* [6, 15, 27] (în engleză "bathtub") ca în Fig.1.4. Punctele a_1, a_2 din figură se numesc *puncte de schimbare* (în engleză "change points"). Estimarea lor se poate face folosind noţiuni de regresie liniară simplă combinate cu teste de comparaţie a coeficienţilor de regresie [29].

1.6.2 Calculul fiabilităţii sistemelor

Cunoscând fiabilităţile $R_i = R_i(t)$ ale componentelor C_i, $1 \leq i \leq n$ se pune problema calculului fiabilităţii $R = R(t)$ a sistemului în funcţie de acele fiabilităţi. *Regula* de calcul este simplă şi ea este dată de următoarea formulă (când sistemul şi componentele au stări binare ca mai sus):

$$R = \Phi(R_1, R_2, ..., R_n)$$

adică fiabilitatea sistemului se calculează cu ajutorul funcţiei Φ în care argumentele x_i se inlocuiesc cu fiabilităţile $R_i, 1 \leq i \leq n$.

In cazul sistemelor serie sau paralel fiabilitatea este respectiv [3]

$$R = \prod_{i=1}^{n} R_i, \quad R = 1 - \prod_{i=1}^{n}(1 - R_i).$$

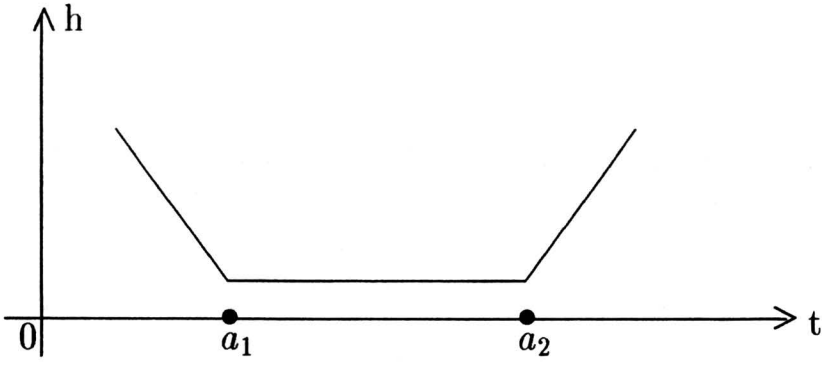


Fig. 1.4. Forma de *cadă de baie a lui h(x).*

In cazul sistemului punte din Fig.1.2c, formula de calcul a fiabilităţii este [3, 6]

$$R = R_5[1-(1-R_1)(1-R_3)][1-(1-R_2)(1-R_4)]+(1-R_5)[1-(1-R_1R_2)(1-R_3R_4)].$$

In general fiabilitatea se poate calcula şi prin analogie cu forma pivotală [3, 23] a funcţiei de structură Φ in raport cu componenta C_i (ca pivot), adică

$$R = R_i R^+_{S_i} + (1 - R_i) R^-_{S_i}$$

unde $R^+_{S_i}$ este fiabilitatea sistemului când componenta C_i funcţionează iar $R^-_{S_i}$ este fiabilitatea sistemului când componenta C_i nu funcţionează. Acest procedeu de calcul poate fi aplicat recursiv pentru subsistemele S_i^+, S_i^- până când se ajunge la forma finală a funcţiei de fiabilitate R a sistemului.

1.6.3 Observaţii finale

(1) Din cele de mai sus rezultă că dacă se cunosc selecţii asupra duratelor de viaţă ale componentelor unui sistem. atunci se pot estima parametrii repartiţiilor duratelor de viaţă (în ipoteza că *tipul* repartiţiei este cunoscut), iar apoi cu ajutorul celor precizate în secţiunile precedente se pot calcula fiabilităţile componentelor sistemului $\overline{F_i}(x) = R_i(x)$, $1 \leq i \leq n$. In continuare, folosind consideraţiile din această secţiune se calculează fiabilitatea sistemului (de ex. pe baza funcţiei de structură).

Aceste calcule de fiabilitate se pot face în *faza de proiectare* a sistemului, putându-se astfel decide apriori dacă sistemul va avea o fiabilitate acceptabilă.

In cazul când sistemul real este deja în funcţiune, calculul fiabilităţii conform tehnicii de mai sus va permite să se decidă dacă sistemul are *încă* fiabilitatea dorită. In cazul când fiabilitatea nu mai este acceptabilă, atunci se vor selecta componentele cu cea mai mare importanţă structurală şi se vor studia mijloace tehnice de creştere a fiabilităţii acelor componente, astfel încât dacă acestea ar fi introduse în sistem să conducă la o fiabilitate mai mare a sistemului.

(2) Procedeele de calcul al fiabilităţii sistemelor pot fi transpuse cu uşurinţă la calculul fiabilităţii programelor. Se ştie (vezi şi Cap. 6) că orice program, conform *teoremei lui Böhm-Jaccopini*, poate fi reprezentat ca un program structurat, descris recursiv în termeni de trei structuri elementare de program şi anume: *secvenţa* $\Pi(A, B)$ care înseamnă că se execută blocul funcţional A urmat de execuţia blocului B; *selecţia* $\Delta(\alpha; A, B)$ care înseamnă *if* α *then* A *else* B, (adică dacă predicatul α ia valoarea *true* atunci se execută blocul A altfel se execută B); iteraţia $\Omega(\alpha, A)$ care înseamnă *while* α *then* A (adică atâta timp cât α ia valoarea *true* se execută blocul funcţional A); în acest ultim caz se presupune că $A = A(\alpha)$, adică α se modifică după un anumit număr de ciclări pentru a determina ieşirea din ciclu. Este uşor de observat că utilizând convenţii potrivite, *orice program structurat poate fi privit ca un sistem SP-generalizat.* Deci fiabilitatea unui program structurat se poate calcula în funcţie de fiabilităţile modulelor componente $A, B, \alpha, etc.$ ţinând seama de următoarele convenţii:

- pentru un predicat α se folosesc probabilităţile

$$p_{t,\alpha} = Prob(\alpha = true), \quad p_{f,\alpha} = Prob(\alpha = false);$$

- pentru secvenţa $\Pi(A, B)$ fiabilitatea este

$$R_\Pi = R_A R_B;$$

- pentru selecţia $\Delta(\alpha, A, B)$ fiabilitatea este

$$R_\Delta = p_{t,\alpha} R_A + p_{f,\alpha} R_B;$$

- pentru iteraţia $\Omega(\alpha, A)$ fiabilitatea este

$$R_\Omega = p_{t,\alpha} \prod_{\alpha=true} R_A + p_{f,\alpha} R_U$$

unde produsul $\prod_{\alpha=true} R_A$ înseamnă că R_A se înmulţeşte cu el însuşi de un număr finit de ori (atâta timp cât $\alpha = true$), iar R_U este fiabilitatea modulului ce urmează imediat structurii de program Ω.

O astfel de abordare ar necesita în primul rând cunoaşterea fiabilităţilor blocurilor funcţionale (modulelor A, B, etc) şi a probabilităţilor predicatelor. Pe de altă parte, dacă ţinem seama că pe parcursul testării programului fiecare eroare detectată se elimină, atunci fiabilitatea programului ar trebui recalculată după eliminarea fiecărei erori. In acelaşi timp acest mod de a aborda fiabilitatea unui produs soft nu ar ajuta la estimarea unei *caracteristici de fiabilitate foarte importantă şi anume, N_0, numărul iniţial de erori.* Acest N_0 permite şi calculul numărului $\overline{N}_0(t)$ de erori rămase la momentul t, care este dat de $\overline{N}_0(t) = N_0 - N(t)$, unde $N(t)$ este numărul de erori eliminate până la momentul t.

Din aceste motive, modelele de fiabilitatea programelor prezentate în capitolele următoare nu se vor concentra numai pe calculul fiabilităţii unui program în funcţie de fiabilităţile componentelor sale. Aceste modele, îşi propun de regulă, ca pe baza unor date de selecţie privind *duratele dintre căderi,* să estimeze pe N_0, numărul mediu de erori rămase la momentul t, $E[\overline{N(t)}]$, precum şi pe $\overline{N(t)} = N_0 - N(t)$, sau alte caracteristici de fiabilitate ce vor fi precizate la locul potrivit în capitolele următoare.

2 Modele Markoviene pentru fiabilitatea programelor

Vom presupune că programele sunt *sisteme reparabile* [2,19,29] în sensul că atunci când apar erori în funcţionarea programelor (erori *soft* ci nu erori cauzate de componente hardware!), atunci aceste programe se întrerup din execuţie şi printr-un proces de depanare, se înlătura erorile. Se presupune că timpul de *depanare* a unei erori este neglijabil în comparaţie cu timpul de execuţie dintre apariţia a două erori consecutive. Se presupune că în program există la momentul iniţial un număr de erori N_0. Pe măsură ce apar noi erori, acestea se elimină (prin depanarea programului) şi deci numărul de erori rămase descreşte. De la început vom presupune că "timpul" în care evoluează programul este *timpul operaţional* (sau timpul *unitate centrală*), adică timpul cumulat pe care-l parcurge programul în execuţie. (El se mai numeşte şi *timp de execuţie*).Aceasta deoarece timpul calendaristic, ce "curge" de la "naşterea" programului nu este relevant ci numai timpul cât programul este utilizat, în execuţie; numai pe parcursul acestui timp putem detecta erorile din program, care de fapt ne interesează.

Inainte de a trata modele de fiabilitatea programelor, trebuie să precizăm *tipurile de date de selecţie* care se folosesc în fiabilitatea programelor. Dacă X este durata în funţionare a unui program sau sistem de programe, atunci datele de selecţie pot fi momentele $0 < T_1 < T_2 < ... < T_n < \infty$ de pe axa timpului operaţional când au loc intreruperile (căderile programului), iar valorile de selecţie asupra lui X sunt $t_1 = T_1, t_i = T_i - T_{i-1}, 2 \leq i \leq n$. In acest caz se spune că avem de-a face cu *date de selecţie complete*. Alt tip de selecţie utilizată în fiabilitatea programelor este *selecţia cenzurată pe intervale*, care se obţine astfel: se aleg constantele pozitive $0 < s_1 < s_2 < ... < s_k, T_n < s_k, k < n$ şi se determină frecvenţa n_j =numărul de căderi ce au loc în intervalul de timp $[s_{j-1}, s_j), n = \sum_{j=1}^{k} n_j$. Valorile $n_1, n_2, ..., n_k$ constituie *datele de selecţie cenzutare pe intervale* ce se vor folosi la estimarea parametrilor modelelor de fiabilitatea programelor. Unele din frecvenţele $n_i, 1 \leq i \leq k$ pot fi nule ceeace înseamna ca pe aceel interval de rang i nu au avut loc căderi (sau mai precis, nu au apărut erori în execuţia programului). Să mai observăm că limitele $s_i, 1 \leq i \leq k$ pot fi aleatoare, caz in care selecţia este cenzurată pe intervale aleatoare. Dacă cenzura este de tipul I (vezi formula (1.23) din subsecţiunea 1.5.1), valoarea de cenzură C de asemenea

poate fi aleatoare, caz in care cenzura este aleatoare de tipul I. In fiabilitatea programelor se folosesc de regulă date cenzurate de tipul II, cu cenzură pe intervale nealeatoare. Când nu se va specifica faptul că este vorba de date cenzurate, se vor folosi valorile de de selectţie necenzurate $t_i = T_i - T_{i-1}$.

Printre primele modele matematice ale fiabilităţii programelor sunt cele care presupun că numărul de erori $N(t)$ existente la momentul t este un proces Markov. Unul din cele mai vechi modele din această categorie este modelul Jelinski-Moranda [16,25] (J-M) ce va fi prezentat în cele ce urmează.

2.1 Modelul Jelinski-Moranda

. Acest model [16,25,35] se bazează pe următoarele ipoteze:

(a) numărul iniţial de erori N_0, existente in program este un parametru constant, necunoscut, ce trebuie estimat;

(b) o eroare detectată este eliminată instantaneu şi nu se mai introduc alte erori;

(c) intervalele de timp dintre apariţiile de erori consecutive sunt variabile aleatoare independente, repartizate exponenţial (consecinţă a faptului că se presune că $N(t)$ este un proces Markovian);

(d) toate erorile existente în soft contribuie în egală măsură la intensitatea de apariţie a unei noi erori.

Să notăm Φ intensitatea de apariţie a unei erori, intensitate presupusă constantă conform ipotezei (d). Deci la momentul iniţial intensitatea de apariţie a unei erori în execuţia programului este $N_0\Phi$. De aici rezultă că după ce au fost detectate şi eliminate k erori ($k = 1, 2, ...$), au mai rămas $N_0 - k$ erori iar intensitatea apariţiei unei noi erori este funcţia descrescătoare in k, $\Phi(N_0 - k)$.

Modelul îşi propune să estimeze parametrii necunoscuţi Φ şi N_0. Datele de selectie (observaţiile statistice) sunt momentele $T_i, 1 \leq i \leq n \leq N_0$ de pe axa timpului operaţional când apar erorile. Conform ipotezei (c) lungimile intervalelor dintre apariţiile consecutive de erori, adică $t_i = T_i - T_{i-1}, (t_0 = 0)$. sunt independente şi repartizate exponenţial de parametri

$$\lambda(i) = \phi[N_0 - (i-1)], 1 \leq i \leq n_0 \tag{2.1}$$

adică t_i are densitatea de repartiţie dr

$$f(t_i) = \Phi(N_0 - i + 1)e^{-\Phi(N_0 - i + 1)t_i}. \tag{2.1'}$$

In concluzie, dacă $\bar{t} = (t_1, t_2, ..., t_n)$ este vectorul de selecţie dat, estimarea parametrilor Φ, N_0 se va face cu metoda verosimilităţii maxime, pornind de la funcţia de verosimiltate (necenzurată)

$$L(t_1, t_2, ..., t_n) = \prod_{i=1}^{n} \Phi(N_0 - i + 1) e^{-\Phi(N_0 - i + 1)t_i} = \tag{2.2}$$

$$= \phi^n \{\prod_{i=1}^{n} (N_0 - i + 1)\} e^{-\Phi \sum_{i=1}^{n} (N_0 - i + 1)t_i}.$$

Ecuaţiile de verosimilitate sunt

$$\frac{\partial ln L}{\partial N_0} = \sum_{i=1}^{n} \frac{1}{N_0 - i + 1} - \sum_{i=1}^{n} \Phi t_i = 0,$$

$$\frac{\partial L}{\partial \Phi} = \frac{n}{\Phi} - \sum_{i=1}^{n} (N_0 - i + 1)t_i = 0.$$

Acest sistem de ecuaţii (neliniare in Φ, N_0) este dificil de rezolvat. Dacă eliminăm pe Φ din cele două ecuaţii obţinem ecuaţia in N_0

$$\frac{1}{N_0} + \frac{1}{N_0 - 1} + ... + \frac{1}{N_0 - n + 1} = \frac{\sum_{i=1}^{n} t_i}{\sum_{i=1}^{n} (N_0 - i + 1)t_i}. \tag{2.3}$$

Soluţia N_0 a ultimei ecuaţii se poate obţine prin metode numerice. Se arată [16, 19, 35] că soluţia N_0 există şi este unică dacă şi numai dacă

$$\frac{\sum_{i=1}^{n} (i-1)t_i}{\sum_{i=1}^{n} (i-1)} > \frac{\sum_{i=1}^{n} t_i}{n} \tag{2.4}$$

adică

$$\frac{\sum_{i=1}^{n} (i-1)t_i}{\sum_{i=1}^{n} t_i} > \frac{1}{2(n-1)}, \tag{2.4'}$$

sau

$$\frac{\sum_{i=1}^{n} it_i - T_n}{T_n} > \frac{1}{2(n-1)}. \tag{2.4''}$$

Dacă în inegalitatea precedentă se schimbă sensul (adică se ia $<$), atunci se arată că soluţia $N_0 \to \infty$ ceea ce nu are sens.

Intrucât condicţia (2.4) nu este în general satisfăcută, rezultă că modelul J-M nu este realist intrucân nu are soluţie N_0.. De fapt, în practică se constată că erorile nu au toate aceeaşi mărime (adică ipoteza (d) nu este de regulă satisfăcută), acesta constituind un alt motiv de a amenda modelul J-M.

2.2 Modele cu intensitatea apariţiei erorilor descrescătoare (DFI)

Observăm că funcţia intensitste a căderilor $\lambda(i) = \Phi(N_0 - i + 1)$ în cazul modelului J-M este liniară şi descrescătoare în i, fapt ce rezultă din ipoteza (d) combinată cu (b). Este însă natural să presupunem [35] că funcţia $\lambda(i)$ nu descreşte liniar ci mai degrabă este descrescătoare-convexă. Aceasta se poate explica prin faptul că la începutul testării sau utilizării programului se detectează cu precădere erorile "mari", cele ce au şanse mai mari de apariţie, iar ulterior se manifestă cele mai mici, "mai ascunse", care apar mai rar. Putem de asemenea presupune că după ce s-au înlăturat toate erorile intensitstea scade la zero, adică $\lambda(N_0+1) = 0$. Un model în care intensitatea erorilor este descrescătoare se numeşte *model DFI* (prescurtare de la *Decreasing Failure Intensity*) [16,19,29].

Vom prezenta mai întâi forma generală a unui model în care $\lambda(i), 1 \leq i \leq N_0$ este descrescătoare şi cunoscută.

In acest caz, numărul de erori rămase $N(t)$ este un proces Markovian particular, anume, un proces *de deces pur*, $N(t) = 0, 1, ..., N_0$, cu repartiţia de forma

$$P[N(t) = i] = p_i(t), i = 0, 1, ..., N_0, \quad \sum_{i=1}^{n} p_i(t) = 1. \tag{2.5}$$

Din ecuaţiile (1.19),(1.19') se deduc ecuaţiile diferenţiale ale procesului nostru de deces pur (ecuaţiile lui Kolmogorov) şi anume

$$p_0(t) = -\lambda(1)p_0(t), \quad p_i'(t) = -\lambda_{i+1}p_i(t) + \lambda_i p_{i-1}(t), 1 \leq i \leq N_0 - 1 \tag{2.6},$$

$$p_{N_0}'(t) = \lambda(N_0)p_{N_0-1}(t).$$

Condiţiile iniţiale naturale sunt

$$N_0(0) = 0, \quad p_0(0) = 1, \quad p_i(0) = 0, i > 0, \quad \sum_{i=0}^{N_0} p_i(0) = 1. \tag{2.6'}.$$

Este adevărată următoarea teoremă [25,35]

Teorema 2.1. *In condiţiile iniţiale (2.6'), sistemul (2.5) are soluţia*

$$p_0(t) = e^{-\lambda_1(t)}, p_1(t) = \frac{\lambda(1)}{\lambda(2) - \lambda(1)}(e_2 - e_1)$$

$$p_i(t) = \sum_{j=0}^{i} A_j^{(i)} e_j, \quad i < N_0, \quad e_j = e^{-\lambda(j+1)t}, 0 \leq j \leq N_0 \tag{2.7}$$

$$p_{N_0}(t) = -\sum_{j=0}^{N_0-1} A_j^{(N_0-1)} \frac{\lambda(N_0)}{\lambda(j+1)}(e_j - 1)$$

unde

$$A_j^{(i)} = \frac{\lambda(i)}{\lambda(i+1) - \lambda(j+1)} A_j^{(i-1)}, \quad j < i, \quad A_i^{(i)} = -\sum_{j=0}^{i-1} A_j^{(i)} \tag{2.8}$$

Demonstraţie. Pentru $i = 0$ soluţia este imediată. Intr-adevăr, prima ecuaţie se scrie

$$\frac{d}{dt} ln p_0(t) = -\lambda(1), \text{ de unde } p_0(t) = ce^{-\lambda(1)t}$$

iar din condiţia iniţială $p_0(0) = 1$ rezultă $c = 1$, adică prima formulă (2.7).

Pentru $i = 1$ din (2.6) avem:

$$p_1'(t) = -\lambda(2)p_1(t) + \lambda(1)p_0(t)$$

care este o ecuaţie diferenţială cu coeficienţi constanţi şi cu termen liber (adică $p_0(t) = e^{-\lambda(1)t}$). Soluţia generală a ecuaţiei omogene este de forma

$$p_1(t) = c_1(t)e^{-\lambda(2)t}.$$

Funcţia $c_1(t)$ se determină prin metoda variaţiei constantelor, ţinând seama şi de termenul liber, adică,

$$p_1'(t) = c_1'(t)e^{-\lambda(2)t} - \lambda(2)c_1(t)e^{-\lambda(2)t} = -\lambda(2)c_1(t)e^{-\lambda(2)t} + \lambda(1)e^{-\lambda(1)t}.$$

Ultima ecuaţie în $c_1(t)$ are soluţia generală

$$c_1(t) = \frac{\lambda(1)}{\lambda(2) - \lambda(1)} e^{-[\lambda(1)-\lambda(2)]t} + k_1.$$

Din condiţia iniţială $p_1(0) = 0$, se deduce $k_1 = -\frac{\lambda(1)}{\lambda(2)-\lambda(1)}$, de unde

$$p_1(t) = \frac{\lambda(1)}{\lambda(2) - \lambda(1)}(e_0 - e_1)$$

adică formula (2.7) pentru $i = 1$. Pentru i oarecare $(1 < i < N_0,)$ formula se demonstrează prin inducţie. Intr-adevăr să presupunem adevărată (2.7) pentru $i - 1$ cu $A_j^{(i-1)}$ dat de (2.8). Pentru i ecuaţia (2.7) se scrie deci

$$p_i'(t) = -\lambda(i+1)p_i(t) + \lambda(i)p_{i-1}(t),\ p_{i-1}(t) = \sum_{j=0}^{i-1} A_j^{(i-1)} e_j.$$

Procedând analog cazului $i = 1$ rezolvarea se face succesiv prin următoarele calcule

$$p_i(t) = c_i(t)e^{-\lambda(i+1)t},$$

$$p_i'(t) = c_i'(t) - \lambda(i+1)c_i(t) = -\lambda(i+1)e^{-\lambda(i+1)t} + \lambda(i)\sum_{j=0}^{i-1} A_j^{(i-1)} e_j,$$

$$c_i'(t) = e^{\lambda(i+1)t}\lambda(i)\sum_{j=0}^{i-1} A_j^{(i-1)} e^{-\lambda(j+1)t},$$

$$c_i(t) = \lambda(i)\sum_{j=0}^{(i-1)} A_j^{i-1}\frac{1}{\lambda(i+1) - \lambda(j+1)} e^{-[\lambda(j+1)-\lambda(i-1)]} + k_i.$$

Din condiţia iniţială $p_i(0) = 0$, rezultă

$$k_i = -\lambda(i)\sum_{j=0}^{i-1}\frac{A_j^{(i-1)}}{\lambda(i+1) - \lambda(j+1)},$$

de unde

$$p_i(t) = [\lambda(i)\sum_{j=0}^{i-1}\frac{A_j^{(i-1)}}{\lambda(i+1) - \lambda(j+1)} e^{-[\lambda(j+1)-\lambda(i+1)]t} -$$

$$-\lambda(i)\sum_{j=0}^{i-1}\frac{A_j^{(i-1)}}{\lambda(i+1) - \lambda(j+1)}]e^{-\lambda(i+1)t} =$$

$$= \sum_{j=0}^{i-1}\frac{A_j^{(i-1)}\lambda(i)}{\lambda(i+1) - \lambda(j+1)} e^{-\lambda(j+1)t} - \sum_{j=0}^{i-1}\frac{A_j^{(i-1)}\lambda(i)}{\lambda(i+1) - \lambda(j+1)} e^{-\lambda(i+1)t} =$$

$$= \sum_{j=0}^{i-1} A_j^{(i-1)} e_j - A_i^{(i)} e_i = \sum_{j=0}^{i} A_j^{(i)} e_j.$$

Pentru $i = N_0$ (ultima ecuaţie (2.7)) avem

$$p'_{N_0} = \lambda(N_0)p_{N_0-1}(t) = \lambda(N_0)\sum_{j=0}^{N_0-1} A_j^{(N_0-1)}e^{-\lambda(j+1)t}.$$

Soluţia generală este

$$p_{N_0}(t) = -\sum_{j=0}^{N_0-1}\frac{\lambda(N_0)}{\lambda(j+1)}A_j^{(N_0-1)}e^{-\lambda(j+1)t} + k_{N_0}$$

Din condiţia iniţială $p_{N_0}(0) = 0$ rezultă

$$k_{N_0} = \sum_{j=0}^{N_0-1}\frac{\lambda(N_0)}{\lambda(j+1)}A_j^{(N_0-1)}$$

deci

$$p_{N_0} = -\sum_{j=0}^{N_0-1}\frac{\lambda(N_0)}{\lambda(j+1)}A_j^{(N_0-1)}[e_j - 1]$$

care este ultima formula (2.7), ceeace demonstrează teorema.

Dacă se estimează în prealabil N_0, conform modelului J-M, atunci se poate calcula probabilitatea ca până la momentul t să se elimine cel puţin k_0 erori, adică

$$1 - P[N(t) \geq k_0] = 1 - \sum_{j=k_0}^{N_0} p_j(t)$$

ultima sumă fiind probabilitatea ca numărul de erori rămase să fie mai mare decât k_0.

2.3 Modele J-M cu repartiţii DFI particulare

Aşa cum am văzut mai sus, în loc de a presupune în modelul original J-M, că pe parcursul testării unui program fiecare eroare existentă în acesta contribuie în aceeaşi măsură la apariţia unei noi erori, este mai potrivit să se presupună că primele căderi sunt cauzate de erori care au o probabilitate mai mare de apariţie. O astfel de presupunere are loc când intensitatea căderilor este dată de funcţia putere

$$\lambda(i) = \Phi[N_0 - (i-1)]^\alpha, 1 \leq i \leq N_0 \qquad (2.9)$$

iar faptul că această funcţie de i descreşte mai repede la început înseamna că ea este convexă, deci $\alpha > 1$. (In cazul modelului clasic J-M tratat anterior, această funcţie este liniară şi descrescătoare in i).

Modelul de tip J-M are, in ipoteza *funcţiei putere* a ratei de apariţie a erorii, trei parametri de estimat: $|Phi, N_0$, *si* α. Dar aşa cum vom vedea mai târziu, o valoare potrivită a lui α este $\alpha = 2$, deci modelul va avea tot doi parametri de estimat, N_0 şi Φ, ca şi în cazul modelului JM clasic . Aceşti parametri vor putea fi estimaţi prin metoda verosimilităţii maxime, în mod asemănător modelului tratat in cele ce urmează.

• **Un model DFI particular.** Să considerăm [25,35] funcţia descrescătoare şi convexă pentru intensitatea căderii a i-a de forma

$$\lambda(i) = \Phi[e^{-\beta(N_0-i+1)} - 1], \quad 1 \leq i \leq N_0. \tag{2.10}$$

Această funcţie descreşte la început mai repede decât funcţia putere.

Vom indica o cale de determinare a estimaţiilor (Φ, N_0, β) folosind metoda verosimilităţii maxime pentru parametri ce intervin în expresia intensitatăţii de apariţie a erorilor dată de (2.10). Se observă că $l = Log(L)$, unde cu L se notează functia de verosimilitate, este

$$l = n \log \Phi + \sum_{i=1}^{n} \log [e^{-\beta(N_0-i+1)} - 1] - \Phi \sum_{i=1}^{n} [e^{-\beta(N_0-i+1)} - 1]t_i.$$

Ecuaţiile de verosimilitate conduc la sistemul neliniar

$$\frac{\partial l}{\partial \Phi} = \frac{n}{\Phi} - \sum_{i=1}^{n} [e^{-\beta(N_0-i+1)} - 1]t_i = 0$$

$$\frac{\partial l}{\partial N_0} = -\beta \sum_{i=1}^{n} \frac{e^{-\beta(N_0-i+1)}}{e^{-\beta(N_0-i+1)} - 1} + \beta\Phi \sum_{i=1}^{n} e^{-\beta(N_0-i+1)} t_i = 0$$

$$\frac{\partial l}{\partial \beta} = -\sum_{i=1}^{n} \frac{(N_0 - i + 1)e^{-\beta(N_0-i+1)}}{e^{-\beta(N_0-i+1)} - 1} + \Phi \sum_{i=1}^{n} (N_0 - i + 1)t_i e^{-\beta(N_0-i+1)}$$

Notând $\mathbf{X} = (\Phi, N_0, \beta)'$ vectorul tridimensional al parametrilor necunoscuţi, sistemul neliniar precedent se pune sub forma

$$\mathbf{X} = \phi(\mathbf{X}) \tag{2.11}$$

unde ϕ este funcţie vectorială. De exemplu din prima şi a treia ecuaţie obţinem respectiv

$$\Phi = n\left[\sum_{i=1}^{n}[e^{-\beta(N_0-i+1)}-1]t_i\right]^{-1}$$

$$N_0 = \frac{\sum_{i=1}^{n}(i-1)t_i e^{-\beta(n_0-i+1)} - \sum_{i=1}^{n}((i-1)e^{-\beta(N_0-i+1)})/(e^{-\beta(N_0-i+1)}-1))}{n[\Phi\sum_{i=1}^{n}t_i e^{-\beta(N_0-i+1)} - \sum_{i=1}^{n}1/(e^{-\beta(N_0-i+1)}-1)]}$$

iar dacă în a doua, simplificăm mai intâi cu β şi adăugăm apoi β in fiecare membru, adică

$$\beta = \beta - \sum_{i=1}^{n}\frac{s^{-\beta(N_0-i+1)}}{E^{-\beta(N_0-i+1)}-1} + \Phi\sum_{i=1}^{n}e^{-\beta(N_0-i+1)t_i},$$

obţinem şi cea de-a treia ecuaţie care conduce la sistemul sub forma (2.11). Pentru un astfel de sistem, când are soluţie, aceasta se determină prin relaţia iterativă

$$\mathbf{X_{n+1}} = \phi(\mathbf{X_n}), \mathbf{X_0} - arbitrar \qquad (2.11')$$

care se repetă până când $\mathbf{X_{n+1}}$ se stabilizează, adică $|\mathbf{X_{n+1}} - \mathbf{X_n}| < \epsilon$, (cu ϵ mic, dat). Soluţia aproximativă

$$\hat{\mathbf{X}} = (\mathbf{X_{n+1}} + \mathbf{X_n})/2$$

poate constitui vectorul estimaţiilor celor trei parametri Φ, N_0, β.

2.4 Justificarea unor ipoteze ale modelului J-M

Ipotezele folosite in cazul modelului Jelinski-Moranda pot fi justificate pe baza unor presupuneri intuitive plausibile.

Mai întâi vom arăta cum se justifică *ipoteza exponenţialităţii,* adică presupunerea că intervalul de timp dintre două căderi are repartiţie exponenţială. Vom începe cu câteva proprietăţi ale repariţiei exponenţiale [6,12].

(1) Am văzut în capitolul 1 că intervalul de timp dintre două evenimente (tranziţii) consecutive ale unui lanţ Markov este repartizat exponenţial. In particular, intervalul de timp τ dintre apariţiile a două evenimente aleatoare rare consecutive ce apar cu intensitatea λ, (al căror număr pe $[0,t]$ este

repartizat Poisson(λ)), este o variabilă exponenţială de parametru λ. Intr-adevăr, dacă $N(t)$ este numărul acestor evenimente , atunci

$$p[N(t)=0]=\frac{(\lambda t)^0}{0!}e^{-\lambda t}=e^{-\lambda t}$$

de unde

$$P(\tau<t)=1-P[N(t)=0]=1-e^{-\lambda t}.$$

(2) Densitatea de repartiţie exponenţială este *strict descrescătoare*; mai mult chiar, dacă ξ este variabila exponenţială $Exp(\lambda)$ şi $t,\Delta t>0$, atunci

$$P(0\le\xi\le\Delta t)>P(t\le\xi\le t+\Delta t). \tag{2.12}$$

Aceasta proprietate rezultă din relaţiile

$$P(0\le\xi\le\Delta t)-P(t\le\xi\le t+\Delta t)=1-e^{-\Delta t}+-e^{-(t+\Delta t)}-e^{-t}>$$

$$>1-e^{-\Delta t}+e^{-(t+\Delta t)}-e^{-(t+\delta T)}=1-e^{-\delta t}>0.$$

(3) Repartiţia exponenţială *nu are memorie* (ca şi procesele de tip Markov!) in sensul că $\forall t,t_0>0$,

$$P(0\le\xi\le t)=P(\xi\le t_0+t|\xi>t_0)=1-e^{-\lambda t}. \tag{2.13}$$

Intr-adevăr, avem

$$P(\xi\le t_0+t|\xi>t_0)=\frac{P(t_0\le\xi\le t_0+t)}{P(\xi>t_0)}=\frac{e^{-t_0}-e^{-(t_0+t)}}{e^{-t_0}}=1-e^{-t},t>0.$$

(4) Repartiţia minimului de variabile exponenţiale independente este tot o variabilă aleatoare exponenţială. Intr-adevăr, dacă $\xi_i\sim Exp(\lambda_i),1\le i\le n$, atunci dacă $\xi=\min\limits_{1\le i\le n}\xi_i$ avem

$$P(\xi<t)=1-P(\xi\ge t)=1-P(\xi_1\ge t)...P(\xi_n\ge t)=1-e^{-(\sum_{i=1}^{n}\lambda_i)t}. \tag{2.14}$$

(5) Pentru un Δt apropiat de zero avem

$$P(0\le\xi\le k\Delta t)\approx k\Delta t,$$

care se deduce din dezvoltarea în serie a expresiei

$$P(0\le\xi\le\Delta t)=1-e^{-\Delta t},\quad\Delta t>0$$

(6) Suma de variabile exponenţiale $\xi_i \sim Exp(\lambda_i)$, $1 \leq i \leq n$, este o variabilă Gamma, adică $\zeta = \sum_{i=1}^{n} \xi_i \sim Gamma(\sum_{i=1}^{n} \lambda_i)$. Demonstraţia acestei afirmaţii se poate face utilizând funcţia caracteristică: se calculează mai intâi funcţia caracteristică a lui ζ, $\varphi_\zeta(t) = E(e^{\zeta t})$, (ca produs de funcţii caracteristice ale variabilelor ξ_i independente şi identic-exponenţial repartizate ca $Exp(\lambda_i)$) şi apoi se calculează funcţia caracteristică a variabilei Gamma şi se observă că cele două sunt egale. In cazul $\xi_i \sim Exp(1)$, variabila sumă are parametrul n (intreg) şi se numeşte variabilă *Erlang.*

Să trecem acum la justificarea exponenţialităţii. Ţinând seama de proprietăţile repartiţiei $Exp(\lambda)$, rezultă că *intuitiv* ipoteza exponenţialităţii se justifică. Vom da însă în cele ce urmează o justificare ce rezultă din însăşi particularităţile unui produs soft [35].

Să notăm cu M mulţimea (discretă) a datelor de intrare ale unui program al cărei cardinal este finit $|M| < \infty$ şi fie $M^* \subset M$, mulţimea datelor ce cauzează erori. Intrucât aceste date de intrare *sosesc rar* pentru a intra în execuţia programului putem presupune că ele (ca evenimente) constituie un process $Poisson(\omega)$. Parametrul ω este interpretat ca *intensitatea testării,* căci cu intensitatea ω sosesc datele în execuţia programului. Funcţia de repartiţie a intervalului de timp τ dintre *sosirile* a două erori consecutive este $F(t) = P(\tau < t)$ iar probabilitatea de a nu avea erori pe intervalul $[0, t]$ este

$$\overline{F}(t) = 1 - F(t) = \sum_{j=0}^{\infty} \left(\frac{e^{-\omega t}(\omega t)^j}{j!} \right) \left(\frac{M - M^*}{M} \right)^j \tag{2.15}$$

formulă ce se justifică astfel: primul factor din sumă este probabilitatea de a sosi j date pe intervalul de timp $[0, t)$ (probabilitatea Poisson), iar al doilea factor înseamna probabilitatea ca cele j date să nu conţină erori; insumarea reprezintă o mediere în raport cu valorile aleatoare ale lui j.

Dacă notăm acum $\lambda = (M^*/M)\omega$ şi interpretăm aceasta ca ceea ce este, adică intensitatea de apariţie a unei erori, atunci (2.15) devine

$$\overline{F}(t) = e^{-\omega t} \sum_{j=0}^{\infty} \frac{[\omega t(1 - \lambda/\omega)]^j}{j!} = e^{-\omega t} e^{\omega(1-\lambda/\omega)t} = e^{-\lambda t}. \tag{2.15'}$$

Ultima formulă spune că $\tau \sim Exp(\lambda)$, adică exponenţialitatea este justificată.

Să vedem acum de unde rezultă în modelul J-M că $\lambda(i) = \Phi(N_0 - i + 1)$.

Să notăm [25,35] M_k numărul datelor de intrare ce pot cauza cea de-a k-a eroare. Atunci, după ce s-a înlăturat cea de-a i-a eroare, numărul datelor de intrare ce pot cauza o nouă cădere este

$$M^*(i) = \sum_{j=i+1}^{N_0} M_j,$$

presupunând că mulţimile M_j ce determină erorile $j = 1, 2, ..., N_0$ sunt disjuncte. Observăm că $M^*(i)$ este descrescătoare in i. Dacă presupunem ca în modelul $J - M$ că toate erorile în program *au aceeaşi mărime,* adică, mulţimile M_k sunt *identice*, $M_k = M_0, k = 1, 2, ..., N_0$, atunci rezultă că numărul de date de intrare ce cauzează o nouă cădere este redus de fiecare dată cu aceeaşi constantă M_0 (notată anterior cu Φ). De aici rezultă că $\lambda(i) = \Phi(N_0 - i + 1)$, adică intervalul de timp t_i dintre a $(i-1)$−a şi a i−a cădere este $Exp(\Phi(N_0 - i + 1))$.

Să justificăm acum ipoteza că rata DFI descrescătoare poate fi de forma funcţiei *putere* considerată în secţiunea 2.3. In acest caz presupunem (in mod mai realist!) [25,35], că erorile *mai "mari"* se elimină mai la începutul testării programului.

Să notăm

$$\overline{M}_i = \frac{M_i^*}{N_0 - i}$$

numărul mediu de date ce pot cauza erorile rămase. Vom presupune acum că numărul de date M_i ce cauzează cea de-a i−a eroare este *de două ori* mai mare ca numărul mediu de date rămase ce pot cauza erori. De aici rezultă că numărul de date ce vor cauza erori după ce s-a eliminat cea de-a $(i+1)$−a eroare este

$$M_{i+1}^* = M_i^* - M_{i+1} = \sum_{j=i+1}^{N_0} M_j - 2\overline{M}_i =$$

$$= \sum_{j=i+1}^{N_0} M_j - \frac{2}{N_0 - i} \sum_{j=i+1}^{N_0} M_j = \frac{N_0 - i - 2}{N_0 - i} \sum_{j=i+1}^{N_0} M_j.$$

De aici se deduce următoarea relaţie de recurenţă

$$\lambda(i+1) = \frac{M_{i+1}^*}{M}\omega = \frac{N_0 - i - 2}{N_0 - i}\frac{\omega}{M}M_i^* =$$

$$= \frac{N_0 - i - 2}{N_0 - i}\lambda(i) = (N_0 - i - 2)(N_0 - i - 1)\frac{\lambda(0)}{(N_0 - 1)N_0}.$$

Pentru N_0 mare, făcând o notaţie corespunzătoare, rezultă

$$\lambda(i+1) \approx (N_0 - i)^2\Phi, \text{ sau } \lambda(i) = (N_0 - i + 1)^2\Phi$$

care este expresia intensităţii putere.

Dacă am face ipoteza că numărul de date M_i ce cauzează cea de-a $i-$a eroare este de α ori numărul mediu al datelor rămase ce pot cauza erori, atunci, printr-un raţionament asemănător se găseşte forma generală a intensităţii "putere" de apariţie a celei de-a $i-$a erori şi anume:

$$\lambda(i) = (N_0 - i + 1)^\alpha\Phi,$$

adică este de forma (2.9).

2.5 Modelul lui Schick-Wolverton

Acest model [1,35] foloseşte ipotezele modelului J-M cu excepţia faptului că repartiţia timpului t_i dintre căderi consecutive este nu exponenţială, ci *este de tip Rayleigh* (un caz particular de repartiţie Weibull) şi anume:

$$f(t_i) = \Phi(N_0 - i + 1)t_i e^{-\frac{1}{2}\Phi(N_0 - i + 1)t_i^2}. \tag{2.16}$$

De aici rezultă că *rata căderilor*, notată aici cu $r(i, t_i)$, este $r(i, t_i) = \Phi(N_0 - i + 1)t_i$, iar *funcţia fiabilitate* este

$$R(t_i) = e^{-\Phi(N_0 - i + 1)t_i^2/2}. \tag{1.16'}$$

Dacă o cădere s-a realizat la momentul t_{i-1} atunci *timpul mediu până la realizarea următoarei căderi* este

$$MTTF(i) = \int_0^\infty f(t_i)t_i dt_i = \Phi(N_0 - i + 1)\int_0^\infty t_i^2 e^{-\Phi(N_0 - i + 1)t_i^2/2}dt_i$$

care după o integrare prin părţi şi după efectuarea schimbării de variabilă $z_i = \sqrt{\Phi(N_0 - i + 1)}t_i$ devine

$$MTTF(i) = \int_0^\infty \frac{1}{\sqrt{\Phi(N_0 - i + 1)}}e^{-t_i^2/2}dt_i = \left[\frac{\pi}{2\Phi(N_0 - i + 1)}\right]^{1/2}. \tag{2.16''}$$

Trebuie să subliniem faptul [35] că ipoteza conform căreia intensitatea (rata) căderilor depinde de timp nu este o presupunere bună. Această ipoteză (a modelului Schick-Wolverton!) conduce la un model realist dacă timpul dintre căderi consecutive t_i se referă la *căderi hardware combinate eventual cu căderi software.*

Rezolvarea modelului Schick-Wolverton se face conform procedeului standard bazat pe metoda verosimilității maxime. Funcţia de verosimilitate este

$$L(t_1, t_2, ..., t_n, N_0, \Phi) = \left(\prod_{i=1}^{n} \Phi(N_0 - i + 1) t_i \right) e^{-(1/2)\sum_i \Phi(N_0 - i + 1) t_i^2}.$$

După logaritmare, notând $l = \ln L$, obţinem *sistemul ecuaţiilor de verosimilitate*

$$\frac{\partial l}{\partial \Phi} = \frac{n}{\Phi} - \frac{1}{2} \sum_{i=1}^{n} (N_0 - i + 1) t_i^2 = 0$$

$$\frac{\partial l}{\partial N_0} = \sum_{i=1}^{n} \frac{1}{N_0 - i + 1} - \frac{\Phi}{2} \sum_{i=1}^{n} t_i^2 = 0.$$

Dacă notăm

$$T^* = \sum_{i=1}^{n} i t_i^2, \quad t^* = \sum_{i=1}^{n} t_i^2$$

atunci din prima ecuaţie se deduce

$$N_0 + 1 = \frac{2n + T^* \Phi}{t^* \Phi} \tag{2.17}$$

iar cea de-a doua devine

$$\sum_{i=1}^{n} \frac{1}{2n + \Phi(T^* - it^*)} = \frac{1}{2}. \tag{2.17'}$$

Ecuaţia (2.17') are intotdeauna soluţie în Φ pe intervalul $[0, 1/(2n)]$ aşa cum se poate vedea uşor. (Deoarece $T^* - it^* > 0, 1 \leq i \leq n$, notând cu $\Psi(\Phi)$ funcţia din membrul stâng al lui (2.17') se deduce uşor că $\Psi'(\Phi) < 0$). Deci, notând cu $\hat{\Phi}$ soluţia ecuaţiei (2.17') (estimaţia de verosimilitate maximă a lui Φ), din (2.17) se poate determina $\hat{N}_0$, estimaţia lui N_0.

2.6 Modelul Markovian al lui Shanthikumar

Modelul este asemănător celui din secţiunea 2.2. Vom presupune [1,26,35] că $N(t)$ este numărul de erori detectate şi eliminate pe intervalul $[0,t]$ iar N_0 este numărul erorilor iniţiale. Să notăm cu $\Phi(t)$ intensitatea unei erori rămase la momentul t. (In cazul modelului J-M, $\Phi(t) = \Phi = const$). Deci dacă s-au eliminat n erori, atunci intensitatea de apariţie a unei noi erori este

$$\lambda(n,t) = \phi(t)(N_0 - n).$$

Rezultă că procesul stochastic $N(t)$ este un proces de naştere pur, care ia valori intre 1 şi N_0. Pentru a afla probabilităţile $P_n(t)$, $t \geq 0$, $1 \leq n \leq N_0$ vom folosi sistemul ecuaţiilor diferenţiale ale lui Kolmopgorov şi anume

$$P_0'(t) = -N_0\Phi(t)P_0(t)$$

$$P_n'(t) = (N_0 - n + 1)\Phi(t)P_{n-1}(t) - (N_0 - n)\Phi(t)P_n(t),\ 1 \leq n \leq N_0 \quad (2.18)$$

care are condiţiile iniţiale

$$P_n(0) = 0,\ n > 0,\ P_0(0) = 1. \quad (2.18')$$

Are loc următoarea teoremă

Teorema 2.2. *Sistemul de ecuaţii diferenţiale (2.18) cu condiţiile iniţiale (2.18') are soluţie unică de forma*

$$P_n(t) = C_{N_0}^n[a(t)]^{N_0-n}[1 - a(t)]^n, \quad 0 \leq n \leq N_0,\ a(t) = e^{-\int_0^t \Phi(t)dt}. \quad (2.19)$$

Demonstraţia se face asemănător celei din secţiunea 2.2. Se observă că după apariţia celei de-a n-a erori expresia funcţiei fiabilitate este

$$R(t) = e^{-(N_0-n)\int_0^t \Phi(u)du} = [a(t)]^{N_0-n}. \quad (2.19')$$

Formula (2.19) este cea a repartiţiei binomiale de parametri $(a(t), n)$ din care cauză modelul se mai numeşte *de tip binomial* [25].

Să mai observăm că dacă ne interesează fiabilitatea la momemtul t ulterior momentului T_{i-1} al ultimei căderi, aceasta este:

$$R(t|T_{i-1}) = e^{-(N_0-i+1)\int_{T_i}^{T_i+t} \Phi(u)du} \quad (2.20)$$

iar rata căderilor la momentul $t \in [T_{i-1}, T_i)$ este

$$\lambda(i, t|T_{i-1}) = (N_0 - i + 1)\Phi(T_{i-1} + t). \qquad (2.20')$$

Se deduce din cele de mai sus că funcţia de repartiţie a variabilei aleatoare T_i (adică a momentului de timp când apare căderea sau eroarea a i-a) este

$$P(T_i < x) = P[N(x) \geq i] = \sum_{j=i}^{N_0} P_j(x) = \sum_{j=i}^{N_0} [a(x)]^{N_0-j}[1 - a(x)]^j. \qquad (2.21)$$

Dacă notăm cu $U(t) = N_0 - N(t)$ numărul de erori rămase la momentul t atunci probabilitatea ca numărul de erori rămase să fie q este

$$P[U(t) = q] = C_{n_0}^q [a(t)]^q [1 - a(t)]^{N_0-q}. \qquad (2.22)$$

Dacă însă ne interesează probabilitatea condiţionată $P[U(t) = q|N(t) = m_e]$ unde $t > t_e$ şi $N(t_e) = m_e$, atunci din (2.19) rezultă că

$$P[U(t) = q|N(t) = m_e] = C_{N_0-m_e}^q [a(t|t_e)]^q [1 - a(t|t_e)]^{N_0-m_e-q},$$

$$a(t|t_e) = e^{-\int_{t_e}^{t} \Phi(u)du}. \qquad (2.22')$$

Datorită formei binomiale a probabilităţii (2.22) rezultă că numărul mediu de erori rămase la momemtul $t, \nu(t)$ şi dispersia acestuia sunt respectiv [25,35]

$$\nu(t) = E[U(t)] = N_0 a(t), \; Var[U(t)] = N_0 a(t)[1 - a(t)]. \qquad (2.22'')$$

2.7 Alte generalizări ale modelului J-M

Ipotezele modelului J-M pot fi uşor modificate pentru a produce noi modele. Vom prezenta in cele ce urmează două astfel de modele.

• **Model cu erori detectate aleator** [25,35]. Desigur, toate erorile de execuţie se detectează şi *se elimină* mai mult sau mai puţin aleator. Aici vom presupune că dacă s-au detectat $i - 1$ erori, acestea s-au eliminat cu probabilitatea $p, 0 < p < 1$, adică numărul mediu de erori eliminate după detectarea celei de-a $(i-1)$-a erori este $p(i-1)$. In acest caz, rata de apariţie a erorii (rata căderilor), menţinând celelalte ipoteze ale modelului J-M, este

$$\lambda(i) = \Phi[N_0 - p(i-1)] = p\Phi[\frac{N_0}{p} - (i-1)]. \qquad (2.23)$$

Rezolvarea acestui model se va face exact ca cea a modelului J-M original dacă se fac în prealabil notaţiile

$$\Phi^* = p\Phi, \quad N_0^* = \frac{N_0}{p}. \tag{2.23'}$$

Parametri (Φ^*, N_0^*) se estimează deci din modelul J-M dedus prin transformările (2.23') adică se obţin estimaţiile $\hat{\Phi}^*$, $\hat{N}_0^*$. Dar în acest fel, parametrul p rămâne neestimat. El se poate estima fie prin altă metodă, fie aplicând direct metoda verosimilităţii maxime. Funcţia de verosimilitate este

$$L(t_1, t_2, ..., t_n; N_0, \Phi, p) = \prod_{i=1}^{n} \left[\Phi(N_0 - p(i-1))e^{-\Phi[N_0 - p(i-1))]t_i}\right] \tag{2.24}$$

care conduce la următorul sistem al ecuaţiilor de verosimilitate

$$\frac{\partial \ln L}{\partial \Phi} = \frac{n}{\Phi} - \sum_{i=1}^{n}[N_0 - p(i-1)]t_i = 0$$

$$\frac{\partial \ln l}{\partial N_0} = \sum_{i=1}^{n} \frac{1}{N_0 - p(i-1)} - \Phi \sum_{i=1}^{n} t_i = 0 \tag{2.24'}$$

$$\frac{\partial l}{\partial p} = \sum_{i=1}^{n} \frac{i-1}{N_0 - p(i-1)} + \Phi \sum_{i=1}^{n}(i-1)t_i = 0.$$

Acest sistem neliniar în cele trei necunoscute N_0, Φ, p, se poate rezolva numeric asemănător sistemului obţinut pentru modelul Schick-Wolverton.

• **Model bazat pe coeficientul de expunere la aroare.** In acest caz parametrul Φ este interpretat ca un *coeficient de expunere la eroare* (FEC=Failure Exposure Coefficient) [25] care depinde de numărul de erori eliminate j. Deci în expresia intensităţii erorilor vom avea $\Phi = k(j)$, iar repartiţia exponenţială a timpilor dintre două căderi consecutive va avea parametrul

$$\lambda(j) = k(j)(N_0 - j). \tag{2.25}$$

Se presupune că FEC $k(j)$ este o funcţie liniară in j de forma

$$k(j) = k_i + \frac{k_f - k_i}{N_0} j, \tag{2.25'}$$

unde k_i şi k_f sunt respectiv FEC-iniţial şi FEC-final, care sunt parametri necunoscuţi împreună cu N_0. Notând $a = k_f - k_i$, rezultă

$$\lambda(j) = k_i N_0 + aj - \frac{a}{N_0} j^2,$$

unde parametri necunoscuţi ce trebuie estimaţi sunt N_0, k_i şi a.

Funcţia de verosimilitate este în acest caz

$$L(t_1, ..., t_n; n_0, k_i, a) = \prod_{j=1}^{n} (k_i N_0 + aj - \frac{a}{N_0} j^2).e^{-\sum_j (k_i N_0 + aj - a/(N_0) j^2) t_j}$$

şi notând ca de obicei $l = \log L$ obţinem sistemul ecuaţiilor de verosimilitate \$ın N_0, k_i, a

$$\frac{\partial l}{\partial N_0} = \sum_j \frac{k_i + \frac{a}{N_0^2} j^2}{k_i N_0 + aj - \frac{a}{N_0} j^2} - \sum_j \left(k_i + \frac{a}{N_0^2} j^2 \right) t_j = 0$$

$$\frac{\partial l}{\partial k_i} = \sum_j \frac{N_0}{k_i N_0 + aj - \frac{a}{N_0} j^2} - k_i \sum_j t_j = 0 \qquad (2.26)$$

$$\frac{\partial l}{\partial a} = \sum_j \frac{j - \frac{1}{N_0} j^2}{k_i N_0 + aj - \frac{a}{N_0} j^2} - a \sum_j \left(j - \frac{j^2}{N_0} t_j \right) = 0.$$

Acest sistem neliniar în N_0, k_i, a se poate rezolva numeric în mod asemănător celor de mai sus.

2.8 Alte Modele Markov

• **Modelul lui Kremer.** Vom face unele presupuneri privind eliminarea erorilor din program [1,2,25]. Notăm $\lambda(t)$= rata căderii programului determinată de apariţia unei erori la momentul t şi presupunem că probabilitatea de a elimina o eroare este p, probabilitatea de a nu elimina o eroare este q, iar probabilitatea de a elimina incorect o eroare este $r, p + q + r = 1$. Atunci $\overline{N}(t) =$ numărul de erori rămase la momentul t este *un proces de naştere şi deces* cu intensităţile de natalitate $\nu_n(t) = n\nu(t), \mu_n(t) = n\mu(t), \nu(t) = r\lambda(t), \mu(t) = p\lambda(t)$. Ne interesează repartiţia procesului $\overline{N}(t)$ adică $P(\overline{N}(t) = n) = P_n(t)$.

Cu aceste notaţii ecuaţiile lui Kolmogorov se scriu

$$P_0'(t) = \mu(t) P_1(t) \qquad (2.27)$$

$$P_n'(t) = (n-1)\nu(t)P_{n-1}(t) - n[\mu(t)+\nu(t)]P_n(t) + (n+1)\mu(t)P_{n+1}(t), n = 1, \ldots, N_0$$

iar condiţiile iniţiale naturale sunt

$$P_n(0) = \begin{cases} 1, & daca \quad n = N_0 \\ 0, & daca \quad n \neq N_0 \end{cases}.$$

Soluţia sistemului (2.27) este

$$P_0(t) = [\alpha(t)]^{N_0}$$

$$P_n(t) = \sum_{i=0}^{min(N_0,n)} C_{N_0}^i C_{N_0+n-i+1}^{N_0-1} [\alpha(t)]^i [\beta(t)]^{N_0-i} [1-\alpha(t)-\beta(t)]^i, n \geq 1, \tag{2.27'}$$

$$\alpha(t) = 1 - \frac{1}{e^{\rho(t)}}, \quad \beta(t) = 1 - \frac{e^{\rho(t)}}{e^{\rho(t)} + A(t)},$$

$$\rho(t) = (p-r)\int_0^t \lambda(u)du, A(t) = r\int_0^t \lambda(u)e^{\rho(u)}du.$$

Acest model este general, dar aplicabilitatea lui depinde de posibilitatea de estimare a parametrului N_0 şi a parametrilor p şi r precum şi de forma funcţiei $\lambda(t)$ care de asemenea trebuie cunoscută sau estimată.

•**Modelul lui Kubat.** Acest model se referă la *fiabilitatea unui software modular*. Se presupune că produsul software este format din M *module* care trebuie să execute K *task*-uri. Un task poate apela mai multe module şi un modul poate fi apelat de mai multe task-uri. Se presupune că tranziţiile dintre module sunt modelate de un proces Markov. Cunoscând intensitatea căderii α_i a fiecărui modul i, $1 \leq i \leq M$, probabilităţile de tranziţie şi maniera de execuţie a modulelor şi task-urilor, se poate descrie un model Markov din care să rezulte câteva caracteristici de fiabilitate. Din presupunerea că procesul este Markovian rezultă că probabilitatea ca să apară cel puţin o cădere in timpul execuţiei task-ului k când se apelează modulul i este

$$1 - P_i(k) = \int_0^\infty e^{-\alpha_i t} g_{ik}(t)dt, 1 \leq i \leq M, 1 \leq k \leq K, \tag{2.28}$$

unde $g_{ik}(t)$ este densitatea de repartiţie a timpului de execuţie a modulului i de către task-ul k.

Dacă $N_i(k)$ este numărul de apelări a modulului i de către task-ul k, atunci valoarea medie a acestuia este

$$a_i(k) = E[N_i(k)], 1 \leq i \leq M, 1 \leq k \leq K \tag{2.29}$$

şi ea poate fi obţinută prin procedeul cunoscut pentru procesele Markov rezolvând ecuaţia

$$a_i(k) = q_i(k) + \sum_{j=1}^{M} p_{ij}(k)a_j(k), \quad i = 1, ..., M; k = 1, ..., K \tag{2.29'}$$

unde $q_i(k)$ este probabilitatea ca task-ul k să apeleze pentru prima dată modulul i, iar $p_{ij}(k)$ este probabilitatea ca task-ul k să apeleze modulul j după ce a apelat modulul i. Folosind mărimile introduse anterior, rezultă că probabilitatea $\pi(k)$ de a avea cel puţin o cădere când se rulează task-ul k este

$$\pi(k) = 1 - \prod_{i=1}^{M}[1 - P_i(k)]^{a_i(k)}, k = 1, ..., K. \tag{2.30}$$

Pentru sistemul software, intensitatea căderii este

$$\lambda_s = \sum_{k=1}^{K} r_k \pi(k), \tag{2.31}$$

unde r_k este *rata* sosirii (în execuţie) a task-ului k. Acum se poate scrie formula pentru fabilitatea sistemului la momentul t când el a avut ultima cădere la momentul t_e, adică

$$R(t|t_e) = e^{-\lambda_s(t-t_e)}. \tag{2.31'}$$

• **Un model de disponibilitate.** Acest model, datorat lui Trivedi şi Shooman [1,25,35], este un model care se referă la *disponibilitate*. Disponibilitatea unui produs software este probabilitatea ca acel produs să fie *în stare de funcţionare* la momentul t. Sistemul software este privit în acest caz ca un *sistem reparabil;* el se găseşte în starea de funcţionare *starea* 1, sau în starea de ne funcţionare *starea* 0, stare în care este *reparat.* Iniţial (la momentul $t = 0$) se presupune că sistemul este în stare de funcţionare. Problema disponibilităţii se pune când se abordează funcţionarea sistemului hard în combinaţie (sau concomitent) cu funcţionarea software-ului şi este

interesant să se cunoască dacă sistemul se află în stare de funcţionare la un moment "crucial" t, adică este *disponibil* la momentul t. Modelul de care ne ocupăm este de tip Markovian. Se presupune că se dă rata căderilor $\lambda(t)$ care reprezintă frecvenţa cu care au loc trecerile din starea de funcţionare în starea de cădere la momentul t şi de asemenea se cunoaşte funcţia $\mu(t)$ (rata *reparaţiilor*) care reprezintă frecvenţa cu care au loc trecerile din starea de cădere în starea de funcţionare la momentul t. Să notăm de asemenea $p_i(t)$ probabilitatea ca sistemul să se afle in a i—a stare de funcţionare la momentul t, $i = n, n-1, \ldots$ şi să notăm $q_i(t)$ probabilitatea ca sistemul să se afle în a i—a stare de cădere la momentul t, $i = m, m-1, \ldots$ Aceste probabilităţi satisfac sistemul de ecuaţii diferenţiale ale lui Kolmogorov:

$$p_n'(t) = -\lambda(t)p_n(t),$$

$$p_{n-i}'(t) = -\lambda(t)p_{n-i}(t) + \mu(t)q_{m-i+1}(t), i = 0, 1, \ldots; \tag{2.32}$$

$$q_{m-i}'(t) = -\mu(t)q_{m-i}(t) + \lambda(t)p_{n-i}(t), i = 0.1, \ldots$$

Condiţiile iniţiale naturale sunt

$$p_n(0) = 1, \quad p_k(0) = 0, \text{ când } k \neq n.$$

Soluţiile sistemului (2.32) (când $\lambda(t) = \lambda t, \mu(t) = \mu t$) sunt

$$p_{n-i}(t) = \left(\frac{\lambda\mu}{\lambda - \mu}\right) e^{-\lambda\mu} \sum_{j=0}^{i} \frac{t^{i-j}[(-1)^{i+1}c_{ij}e^{-(\mu-\lambda)t} + (-1)^j d_{ij}]}{(\mu - \lambda)(i-j)!}, \tag{2.33}$$

unde

$$c_{ij} = \begin{cases} 0, & daca \quad j = 0 \\ 1, & daca \quad j = 1, \\ C_{i+j-1}^{j-1} & altfel \end{cases} \tag{2.33'}$$

$$d_{ij} = \begin{cases} 1 & daca \quad j = 0 \\ C_{i+j-1}^{j-1} & altfel \end{cases}$$

iar, în mod asemănător

$$q_{m-i} = \frac{1}{\mu}\left(\frac{\lambda\mu}{\mu - \lambda}\right)^{i+1} e^{-\lambda t} \sum_{j=0}^{i} \frac{c_{ij+1}t^{i-j}[(-1)^{i+1}e^{-(\mu-\lambda)t} + (-1)^j]}{(\mu - \lambda)(i-j)!}. \tag{2.34}$$

Disponibilitatea la momentul t este deci

$$A(t) = \sum_{k=0}^{\infty} p_{n-k}(t).$$

Soluţiile de mai sus au fost determinate când $\lambda(t) = \lambda t, \mu(t) = \mu t$, adică sunt funcţii liniare, adică într-un caz simplu. Un alt caz interesant se poate obţine când

$$\lambda(t) = e^{a+br^t}, \quad \mu(t) = 1/(1 + e^{a+br^t}) \tag{2.35}$$

unde a, b, r sunt parametri necunoscuţi.

2.9 Modele cu date cenzurate

Există mai multe moduri de a defini *date cenzurate* [8, 17]. Datele temporale *cenzurate de tipul I* (durate in funcţionare) se caracterizează prin faptul că ele sunt obţinute fie la cădere, fie după ce sistemul a funcţionat până la un anumit *timp de cenzură* T. Deci o oprire a programului necauzată de o eroare într-un experiment nedefinit ca durată poate fi privită ca un *eveniment cenzurat.* Datele cenzurate de tipul II sunt acelea care specifică intervalele de timp $(a_j, a_{j+1}]$, $0 = a_0 < a_1 < a_2 < ... < a_k$ în care au loc erorile şi *frecvenţele* $s_j, 1 \leq j \leq k$ ale acestor erori pe intervalul $(a_j, a_{j+1}], \sum_{j=1}^{k} s_i = n=$ numărul de erori observate pe intervalul $(0, a_n]$. Intervalele de cenzură pot avea limitele $a_j, 1 \leq j \leq k$ constante sau aleatoare, de repartiţie cunoscută. Presupunem că momentele la care se observă opririle programului sunt [2,8,17] T_i şi duratele dintre opriri (pe axa timpului calendaristic) sunt $t_i = T_i - T_{i-1}, T_0 - 0, 1 \leq i \leq n$. Datele cenzurate de tipul I sunt $y_i = t_i \vee T = min\{t_i, T\}, 1 \leq i \leq n$, iar datele cenzurate de tipul II (când intervalele de cenzură sunt $(a_j, a_{j+1}]$), sunt $s_j, 1 \leq j \leq n$.

Vom folosi următoarele notaţii în plus faţă de cele precedente: intervalul de timp $[0, T]$ pe care se *înregistrează* opririle este divizat în intervale $\tau_i = (a_{i-1}, a_i]$. Aceste intervale sunt de forma $(a_i, a_i'] \bigcup (a_i'', a_{i+1}), a_i' < a_i''$ intervalul $(a_i', a_i'']$ fiind interval de *timp de lenevire* când programul *stă* neactivat. Deci momentele de oprire a_i ale programului pot fi momente de oprire *cu cădere* sau momente de oprire *fără cădere.* Fie $b_i = \chi(a_i)$ valoarea *funcţiei indicator* a momentului de timp a_i şi $M_k = \sum_{j=1}^{k} b_j$ numărul opririlor cu cădere până la momentul a_k. Datele sunt *cenzurate* (cenzură de tipul II *cu date grupate*, cu limitele intervalelor de grupare nealeatoare) în sensul că noi

nu putem înregistra exact căderile, ci putem preciza numai intervalele de timp τ_i în care au loc căderile. (Pe un interval $\tau_i = (a_i, a_{i-1}]$ pot să nu aibă loc căderi!. Când opririle ce au loc nu sunt cauzate de erori spunem că suntem în cazul *demonstraţiilor de software*). Datele noastre sunt deci (t_i, b_i), şi sunt presupuse a fi variabile aleatoare independente stochastic.

• **Modelul J-M cenzurat [17].** In ipotezele modelului J-M avem rata căderilor şi funcţia fiabilitate date respectiv de formulele

$$h(t) = h_i = \Phi(N_0 - M_{i-1}),\ t \in \tau_i, \quad R_i(t) = e^{-h_i t}. \tag{2.36}$$

Dacă oprirea i la intervalul de interoprire t_i este o cădere, atunci

$$P(b_i = 1, x_i \leq t_i \leq x_i + dx_i) = f_i(x_i) \tag{2.36'}$$

iar dacă aceasta nu este o cădere, atunci

$$P(b_i = 0, t_i > x_i) = R(x_i). \tag{2.36''}$$

De aici rezultă că funcţia de verosimilitate este

$$L[(b_1, t_1), ..., (b_n, t_n)] = \prod_i [f(t_i)]^{b_i} [R_i(t_i)]^{1-b_i}. \tag{2.37}$$

Din (2.37) rezultă că dacă opririle non-căderi sunt mai multe decât opririle căderi (ceeace este de presupus în practică), atunci funcţia de verosimilitate este dominată de datele cenzurate, deci modelul cu date cenzurate este legitim de considerat. Dacă ţinem seama de (2.36) şi de faptul că $f_i(x)$ în cazul modelului J-M, este densitatea exponenţială de parametru $\Phi(N_0 - M_{i-1})$, atunci logaritmul funcţiei de verosimilitate este

$$l = \sum_i [b_i log(N_0 - M_{i-1}) + b_i log(\Phi) - b_i(N_0 - M_{i-1})\Phi t_i$$

$$-(1 - b_i)(N_0 - M_{i-1})\Phi t_i]. \tag{2.38}$$

Ecuaţiile de verosimilitate sunt

$$\frac{\partial l}{\partial N_0} = \sum_i \left[\frac{b_i}{N_0 - M_{i-1}} - \Phi t_i \right] = 0$$

$$\frac{\partial l}{\partial \Phi} = \sum_i \left[\frac{b_i}{N_0 - M_{i-1}} - \Phi t_i \right] \frac{N_0 - M_{i-1}}{\Phi} = 0$$

care se pot pune sub forma

$$\sum_i \frac{b_i}{N_0 - M_{i-1}} = \Phi T_n$$

$$\frac{d}{\Phi} = N_0 T_n - \sum_i t_i M_{i-1}, \quad d = \sum_i b_i \tag{2.38'}$$

unde d este numărul datelor necenzurate iar T_n este momentul ultimei opriri a programului. Eliminând pe Φ din ecuaţiile (2.38') se obţine o ecuaţie în N_0 care se poate rezolva prin metode numerice iterative.

In cazul *demonstraţiilor de software* (când opririle sunt fără căderi) avem

$$h(t) = h_i = N_0\Phi = h = const.$$

şi deci funcţia fiabilitate este

$$L[(t_1, b_1), ..., (t_n, b_n)] = \prod_i [he^{-ht_i}]^{b_i} . [e^{-ht_i}]^{1-b_i}$$

iar parametrul necunoscut este h. Ecuaţia de verosimilitate pentru h este

$$\frac{\partial log L}{\partial h} = \sum_i [b_i/h - t_i] = 0$$

de unde rezultă estimaţia

$$\hat{h} = \frac{\sum_i b_i}{\sum_i t_i} = \frac{d}{T_n}. \tag{2.39}$$

Ultima formulă rămâne adevărată dacă notăm K_i= numărul de căderi pe intervalul τ_i (limitele a_i sunt extremităţile intervalelor *de cenzură*) şi considerăm $K_i, 1 \leq i \leq k$ variabile aleatoare $Poisson(ht_i)$, independente. Intr-adevăr, funcţia de verosimilitate este în acest caz

$$L[(k_1, t_1), ..., (k_n, t_n)] = \prod_{i=1}^{k} \frac{(h.t_i)^{k_i}}{k_i!} e^{-ht_i}$$

unde n este numărul total al căderilor. Ecuaţia de verosimilitate pentru h ne conduce la estimaţia

$$\hat{h} = \frac{\sum_i k_i}{T_n} = \frac{n}{T_n}.$$

(In ultima formulă avem de fapt $T_n = a_k$). Se observă că în cazul demonstraţiilor de software, modelul J-M cu date cenzurate nu se reduce la cel clasic (fără date cenzurate), acest fapt nefiind convenabil pentru aplicaţiile practice. De aceea vom presupune în loc de (2.36) că

$$h(t) = \alpha + \beta M_{i-1}, t \in \tau_i, i = 1, 2, ... \tag{2.40}$$

iar după construirea funcţiei de verosimilitate se deduc următoarele ecuaţii (din condiţia de maxim a log-verosimilităţii) pentru parametrii α, β

$$\sum_i \left[\frac{b_i}{\alpha + \beta M_{i-1}} - t_i \right] = 0$$

$$\sum_i \left[\frac{b_i}{\alpha + \beta M_{i-1}} - t_i \right] M_{i-1} = 0. \tag{2.41}$$

Soluţiile $\hat{\alpha}, \hat{\beta}$ ale acestui sistem se obţin cu uşurinţă prin metode numerice.

•**Modelul Schick-Wolverton cenzurat [17].** Presupunem că funcţia rata căderilor este de forma

$$h(t) = \Phi[N_0 - M_{i-1}](t - t_{i-1}), t \in \tau_i \tag{2.42}$$

deci funcţia fiabilitate este

$$R_i(t) = P[t_i > t | t_1, ..., t_n] = e^{-\Phi(N_0 - M_{i-1})\frac{1}{2}x^2}.$$

De aici rezultă funcţia de verosimilitate

$$L(t_1, t_2, ..., t_n; \Phi, N_0) = \prod_{i=1}^{n} [(N_0 - M_{i-1}) t_i \Phi]^{b_i} \cdot e^{-\Phi \sum_{i=1}^{n} (1-b_i)(N_0 - M_{i-1}) \frac{t_i^2}{2}} \tag{2.43}$$

iar ecuaţiile de verosimilitate ale parametrilor N_0 şi Φ sunt

$$\sum_{i=1}^{n} \left[\frac{b_i}{N_0 - M_{i-1}} - \frac{1}{2}\Phi t_i^2 \right] = 0$$

$$\sum_{i=1}^{n} \left[\frac{b_i}{N_0 - M_{i-1}} - \frac{1}{2}\Phi t_i^2 \right] \frac{N_0 - M_{i-1}}{\Phi} = 0. \tag{2.44}$$

Ca şi în modelul J-M, şi aici, modelul cu date cenzurate pentru demonstraţii de software nu se reduce la cel clasic (de tip Schick-Wolwreton). De aceea vom modifica ipoteza (2.42) astfel

$$h(t) = \alpha + (\beta + \gamma M_{i-1})(t - t_{i-1}), t \in \tau_i. \tag{2.45}$$

Procedând ca în cazul celorlalte două modele precedente se obţine pentru parametri α, β, γ sistemul de ecuaţii

$$\begin{aligned} &\sum_i \left[\frac{b_i}{\alpha + (\beta + \gamma M_{i-1})t_i} - t_i\right] = 0 \\ &\sum_i \left[\frac{b_i}{\alpha + (\beta + \gamma M_{i-1})t_i} - \frac{1}{2}t_i\right] t_i = 0 \\ &\sum_i \left[\frac{b_i}{\alpha + (\beta + \gamma M_{i-1})t_i} - \frac{1}{2}t_i\right] t_i M_{i-1} = 0 \end{aligned} \tag{2.46}$$

care se rezolvă de asemenea prin metode numerice.

• **Modelul J-M cenzurat de tip geometric [17].** In acest model (nestudiat anterior in cazul necenzurat) se presupune că

$$h(t) = D\Phi^{M_i}, t \in \tau_i, \tag{2.47}$$

unde D, Φ sunt parametri necunoscuţi. Ţinând seama de faptul că funcţia fiabilitate este

$$R_i(t) = P(t_i > t | t_1, ..., t_n) = e^{-D\Phi^{M_i} t},$$

se construieşte funcţia de verosimilitate conform formulei (2.37) care în final conduce la următoarele ecuaţii de vrosimilitate pentru parametrii $D. \Phi$

$$\begin{aligned} &\sum_i \left[\frac{b_i}{D} - \Phi^{M_i} t_i\right] = 0 \\ &\sum_i \left[\frac{b_i}{D} - \Phi^{M_i} t_i\right] M_i = 0. \end{aligned} \tag{2.48}$$

Rezolvarea sistemului neliniar (2.48) se face de asemenea prin metode numerice, iterative. Se observă că in cazul $\hat{h}(\Phi) = 1$ avem $h = constant$, adică suntem în cazul demonstraţiei de software.

• **Modelul cenzurat al lui Littlewood-Verall [17].** Presupunem că funcţia rata căderilor este de forma

$$h(t) = \frac{\alpha}{(t_i - t_{i-1}) + \Psi(i)}, \quad t \in \tau_i, \Psi(i) = \beta + \gamma M_i \tag{2.49}$$

unde τ_i sunt intervalele de grupare. Atunci funcţia fiabilitate cu care se construieşte funcţia de verosimilitate este

$$R_i(t) = P\{t_i > t | t_1, ..., t_n\} = \left[1 + \frac{t}{\Psi(i)}\right]^{-\alpha}. \tag{2.50}$$

Ţinând seama de (2.37) şi logaritmând funcţia de verosimilitate se obţin ecuaţiile de verosimilitate

$$\sum_i \left[\frac{b_i}{\hat{\alpha}} + \log(\hat{\beta}) + \gamma M_i) - \log(t_i + \hat{\beta} + \hat{\gamma} M_i)\right] = 0$$

$$\sum_i \left[\frac{\hat{\alpha}}{\hat{\beta} + \hat{\gamma} M_i} - \frac{b_i + \hat{\alpha}}{t_i + \hat{\beta} + \hat{\gamma} M_i}\right] = 0$$

$$\sum_i \left[\frac{\hat{\alpha}}{\hat{\beta} = \hat{\gamma} m_i} - \frac{b_i + \hat{\alpha}}{t_i + \hat{\beta} + \hat{\gamma} M_i}\right] M_i = 0.$$

Se observă că că soluţia modelului in acest caz revine la $h(t) = 0$. De aceea putem lua $h(t)$ de forma

$$h(t) = \alpha_0 + \frac{\alpha}{t - t_{i-1} + \beta + \gamma M_i}, \quad t \in \tau_i$$

dar şi in acest caz intervine un inconvenient deoarece avem de estimat patru parametri. O alegere mai bună a funcţiei $h(t)$ este

$$h(t) = \alpha + \frac{\beta M_i}{(t - t_{i-1}) + \gamma}, \quad t \in \tau_i.$$

Ecuaţiile de verosimilitate ale modelului cenzurat sunt în acest caz

$$\sum_i \left[\frac{b_i(t_i + \hat{\gamma})}{\hat{\alpha}(t_i + \hat{\gamma}) + \hat{\beta} M_i} - b_i t_i\right] = 0$$

$$\sum_i \left[\frac{b_i(t_i + \hat{\gamma})M_i}{\hat{\alpha}(t_i + \hat{\gamma}) + \hat{\beta}M_i} - \log\left(1 + \frac{t_i|}{\hat{\gamma}}\right) \right] M_i = 0$$

$$\sum_i \left[\frac{b_i}{\hat{\alpha}(t_i + \hat{\gamma}) + \hat{\beta}M_i} - \frac{1}{\hat{\gamma}} + \frac{1}{t_i + \hat{\gamma}} \right] \hat{\beta}M_i = 0.$$

O soluţie trivială a acestui model este

$$\hat{\alpha} = \frac{M_i}{n \sum_i b_i t_i}, \quad \hat{\beta} = 0, \quad \hat{\gamma} = oarecare$$

iar valoarea maximă a funcţiei de verosimilitate este

$$\hat{L} = \hat{\alpha}^{M_n} \exp(-\alpha \sum_i t_i).$$

Se poate intâmpla ca sistemul precedent să admită şi alt punct de extrem. Solucţia cea mai bună este aceea care conduce la o valoare mai mare a funcţiei de verosimilitate.

3 Modele bazate pe procese Poisson neomogene

Aceste modele se construiesc în ipoteza că numărul de căderi $N(t)$ observate pe intervalul $[0,t)$ (adică numărul de erori detectate şi eliminate pe acest interval de timp), este un proces Poisson neomogen $PPNO$ de medie $m(t) = E[N(t)] > 0$. Deci

$$P[N(t) = k] = \frac{[m(t)]^k}{k!} e^{-m(t)}. \; k = 0, 1, 2, ...$$

Dacă notăm $\overline{N}(t) = N(\infty) - N(t)$, numărul de erori *rămase* la momentul t, atunci

$$E[\overline{N}(t)] = m(\infty) - m(t)$$

$$P[\overline{N}(t) = k] = \frac{[m(\infty) - m(t)]^k}{k!} e^{-[m(\infty)-m(t)]}, k = 1, 2, ... \qquad (3.1)$$

Ţinând cont de precizările din Cap.I rezultă că fiabilitatea la momentul $t_0 + t$ condiţionată de faptul că programul nu a căzut pe intervalul $t_0, t_0 + t$ este

$$R(t|t_0) = e^{-[m(t_0+t)-m(t_0)]} \qquad (3.2)$$

O carazcteristică importantă a fiabilităţii unui program este *timpul mediu cumulat dintre căderi la momentul* $t, t \in (0, \infty)$, $CMTBF(t)$ (Cummulative Mean Time Between Failures) care va fi notat $\Psi(t)$. Relaţia între $m(t)$ şi $\Psi(t)$ este

$$\Psi(t) = \frac{1}{m(t)} \qquad (3.2')$$

iar dacă notăm cu $T_1, T_2, ..., T_n, T_i < T_{i+1}$, $1 \leq i \leq n-1$ momentele de timp observate ale căderilor, atunci estimaţia lui $\Psi(t)$ este

$$\hat{\Psi}(t) = \frac{T_n}{n}, \; T_n \leq t < T_{n+1}. \qquad (3.2'')$$

Dacă însă $m(t)$ se estimează pe altă cale (aşa cum se va vedea mai jos) atunci pentru estimarea lui $\Psi(t)$ se poate folosi formula (3.2').

Modelele bazate pe $PPNO$ presupun că datele de selecţie sunt *cenzurate pe intervale,* adică valorile de observaţie sunt $n_1, n_2, ..., n_k$ unde n_i = numărul

de căderi detectate pe intervalul $[s_{i-1}, s_i), s_0 = 0, i = 1, 2, ..., k$, unde $n_1 + n_2 + ... + n_k = n$. (Intervalele de cenzură sunt determinate de punctele date $0 = s_0 < s_1 < ... < s_k$). De aici rezultă că funcţia de verosimilitate este

$$L(n|n_1, n_2, ...n_k) = \prod_{i=1}^{k} \frac{[m(s_{i-1}) - m(s_i)]^{n_i}}{n_i!}]e^{[m(s_i)-m(s_{i-1}]}. \qquad (3.3)$$

In general, funcţia de fiabilitate depinde de parametri necunoscuţi (aşa cum vom vedea mai jos), care se estimează din condiţia $L = max$ (adică se estimează cu metoda verosimilităţii maxime).

Să amintim câteva proprietăţi ale PPNO:

(1).Suma unor procese Poisson $N_i(t), 1 \leq i \leq r$ de parametri $m_i(t)$ este tot un PPNO notat $N(t)$, de parametru $m(t) = \sum_{i=1}^{r} m_i(t)$;

(2). Orice PPNO $N(t)$ se poate transforma intr-un alt PPNO $N^*(t) = N(a(t))$ (adică prin transformarea $t \to a(t)$), având media $m^*(t) = m(a(t))$.

In particular, dacă $m(t)$ este inversabilă, atunci transformarea $a(t) = m^{-1}(t)$ conduce la procesul Poisson $N^*(t) = N(m^{-1}(t))$ care este un proces Poisson *omogen* PPO având media $m(t) = t$ (adică un proces PPO cu intensitatea constantă egală cu 1).

Să amintim că *intensitatea PPNO* este

$$\lambda(t) = \frac{dm(t)}{dt}, \; m(s) = \int_0^s \lambda(t)dt \qquad (3.4)$$

care se mai numeşte şi *rata periculozităţii,* denumire ce provine din demografie.

In cele ce urmează vom prezenta câteva modele partriculare bazate pe $PPNO$, adică modele ce corespund unor forme particulare ale funcţiei $m(t)$.

Funcţia $m(t)$ a fiecărui model depinde de nişte paramatri $\theta_1, ..., \theta_k$ ce se estimează cu metoda verosimilităţii maxime sau cu *metoda celor mai mici pătrate* aşa cum se va vedea mai încolo.

3.1 Modelul lui Goel-Okumoto (GO)

Ipotezele acestui model sunt următoarele [2, 30, 35]:

-Toate erorile sunt independente şi au aceeaşi şansă de apariţie;

-Toate erorile detectate sunt eliminate instantaneu şi nu se mai introduc alte erori;

- Numărul $N(t)$ de erori detectate până la momentul t este un $PPNO$ având media

$$m(t) = a(1 - e^{-bt}), \; a > 0, b > 0 \qquad (3.5)$$

iar $\overline{N}(0)$=numărul inițial de erori este o variabilă aleatoare Poisson.

Din ipotezele de mai sus rezultă

$$\lambda(t) = abe^{-bt},\ m(\infty) = a, m(0) = 0$$

iar numărul mediu de erori rămase la momentul t este 0

$$E[\overline{N}(t)] = m(\infty) - m(t) = ae^{-bt}. \tag{3.6}$$

Dacă ultima cădere a programului a avut loc la momentul s, atunci funcția de fiabilitate condiționată este

$$R(t|s) = e^{-a(e^{-bs}-e^{-b(s+t)})}. \tag{3.6'}$$

Să vedem mai întâi cum se justifică formula (3.5) a mediei $m(t)$ în cazul modelului Goel-Okumoto. Acest fapt rezultă din ipoteza că numărul de erori ce se detectează în intervalul $[t, t+\Delta t)$ este *proporțional* cu numărul de erori rămase, adică

$$m(t+\Delta t) - m(t) = b[a - m(t)].\Delta t \tag{3.7}$$

unde b este o constantă de proporționalitate iar a este numărul mediu de erori inițiale. Din (3.7) rezultă ecuația diferențială

$$m'(t) = b[a - m(t)] \tag{3.7'}$$

a cărei soluție este (3.5) adică chiar media $PPNO$ în cazul modelului GO.

Să vedem acum cum se estimează parametrii a, b ai modelului GO. Funcția de verosimilitate (3.3) este în acest caz

$$L(n_1, n_2, ..., n_k) = \prod_{i=1}^{k} [\frac{[a(e^{-bs_{i-1}} - e^{-bs_i})]^{n_i}.e^{-a[e^{-bs_{i-1}}-e^{-bs_i}]}}{n_i!}. \tag{3.7}$$

Se observă că

$$l = lnL = \sum_{i=1}^{k} n_i ln[e^{-bs_{i-1}} - e^{-bs_i}] - \sum_{i=1}^{k} a[e^{-bs_{i-1}} - e^{-bs_i}] - \sum_{i=1}^{k} ln(n!) + \sum_{i=1}^{k} n_i lna. \tag{3.7'}$$

Din condițiile de minim pentru funcția de verosimilitate rezultă sistemul

$$\frac{\partial l}{\partial a} = \frac{\sum_{i=1}^{k} n_i}{a} - \sum_{i=1}^{k}(e^{-bs_{i-1}} - e^{-bs_i}) = 0$$

$$\frac{\partial l}{\partial b} = \sum_{i=1}^{k} \frac{n_i(s_i e^{-bs_{i-1}} - s_{i-1}e^{-bs_i})}{e^{-bs_{i-1}} - e^{-bs_i}} - as_k e^{-bs_k} = 0.$$

Prima ecuaţie a sistemului se mai scrie

$$a = \frac{n}{1 - e^{-bs_k}}, \; n = \sum_i n_i \tag{3.8}$$

şi eliminând pe a din cea de a doua ecuaţie obţinem

$$\sum_i \frac{n_i(s_i e^{-bs_{i-1}} - s_{i-1}e^{-bs_i})}{e^{-bs_{i-1}} - e^{-bs_i}} - \frac{ns_k}{1 - e^{-bs_k}} e^{-bs_k}. \tag{3.8'}$$

Ecuaţia (3.8') se rezolvă numeric în b obţinându-se estimaţia $\hat{b}$, iar din (3.8) se obţine estimaţia $\hat{a}$ a lui a.

Forma (3.5) a modelului GO se poate generaliza şi anume, considerând

$$m(t) = a[1 - e^{-bt^c}] \tag{3.9}$$

unde a reprezintă numărul mediu de erori existente în program la momentul iniţial, b este un parametru de scală, iar c este un parametru ce caracterizează *calitatea testului* [25] cu care se detectează erorile. Intensitatea căderilor este în acest caz

$$\lambda(t) = \frac{dm(t)}{dt} = abct^{c-1}e^{-bt^c}. \tag{3.9'}$$

3.2 Modele bazate pe $PPNO$ "în formă de S" [25,35].

In modelul GO funcţia $m(t)$ este crescătoare pe $(0, \infty)$ şi convexă. Aceasta se întâmplă datorită ipotezelor ce stau la baza modelului GO, printre care ipoteza că erorile din soft ar fi independente şi "de aceeaşi mărime". Dar de fapt erorile nu sunt nici independente şi nici de aceeaşi mărime sau importanţă. La începutul testării unui soft unele erori sunt *acoperite* de alte erori ce nu sunt încă detectate.Erorile acoperite nu pot fi detectate până nu se înlătură erorile care le acoperă. Deci defectele softului înlăturate la inceput, nu descresc foarte mult intensitatea căderilor deoarece aceleaşi date de test încă pot conduce la căderi cauzate de erorile acoperite.Aceasta conduce la o creştere a ratei căderilor la începutul testării. In schimb mai târziu erorile mari cauzatoare de căderi sunt deja înlăturate şi erorile mai mici pot cauza atunci erori, fapt ce determină o creştere mai lentă a ratei căderilor. Aceste

considerente conduc la ipoteza realistă că funcţia $m(t)$ trebuie să aibă un *grafic în formă de* S, adică la inceput (pentru t mic) funcţia creşte mai rapid având formă convexă, iar mai apoi creşte mai lent având forma concavă.

Altă explicaţie a modelelor "in formă de S" este aceea că testarea presupune un *proces de învăţare* prin care oamenii care testează programe se familiarizează mai întâi cu acestea şi cu tehnica testării, apoi devin eficienţi si pentru un timp detectează mai repede erorile, iar după ce erorile mai importante şi cele "ascunse" sunt înlăturate, erorile rămase se detectează din ce în ce mai rar. Din aceste cauze numărul mediu de erori $m(t)$ are un grafic în forma de S.

• **Un model tipic bazat pe** $PPNO$**, în formă de** S este cel care are media de forma [35]

$$m(t) = a[1 - (1 + bt)e^{-bt}], a > 0, b > 0 \tag{3.10}$$

iar intensitatea căderilor (rata periculozităţii) este

$$\lambda(t) = \frac{dm(t)}{dt} = ab^2te^{-bt} \tag{3.10'}$$

unde a este numărul iniţial de erori ce trebuie detectate iar b este rata de detectare a erorilor.

De aici rezultă numărul mediu de erori rămase în soft la momentul t şi anume

$$\overline{m}(t) = m(\infty) - m(t) = a - a[1 - (1 + bt)e^{-bt}] = a(1 + bt)e^{-bt}. \tag{3.10"}$$

Parametrii a şi b se estimează cu ajutorul metodei verosimilităţii maxime, conform procedeului obişnuit.

• **Modelul lui Shagen [1, 35].** Un alt model în formă de S, datorat lui Schagen, este cel care presupune că media $PPNO$ este

$$m(t) = \alpha[1 - e^{-\lambda_1 t} - \frac{\lambda_2(e^{-\lambda_1 t} - e^{-\lambda_2 t})}{\lambda_2 - \lambda_1}] \tag{3.11}$$

unde $\alpha, \lambda_1, \lambda_2$ sunt parametri ce trebuie estimaţi. Ei se pot estima cu ajutorul metodei verosimilităţii maxime. Se observă că dacă $\lambda_1 = \lambda_2$ (notat cu b) şi $a = \alpha$ se obţine modelul în formă de S anterior.

• **Model cu factor de incovoiere [35].** Insfârşit, un alt model în formă de S consideră funcţia

$$m(t) = \frac{a(1 - e^{-bt})}{1 + ce^{-bt}} \tag{3.12}$$

unde a este numărul de erori ce trebuie detectate, b este rata de detectare a erorilor, iar c este *factorul de încovoiere.*

Din (3.12) rezultă după calcule că intensitatea căderilor este

$$\lambda(t) = \frac{dm(t)}{dt} = \frac{ab(1+c)e^{-bt}}{(1+ce^{-bt})^2} \tag{3.12'}$$

iar numărul de erori rămase este

$$\overline{m}(t) = m(\infty) - m(t) = \frac{a(1+c)e^{-bt}}{1+ce^{-bt}}. \tag{3.12"}$$

Pentru estimarea parametrilor fiecăruia din cele trei modele anterioare se poate aplica metoda verosimilităţii maxime pornind de la expresia generală (3.3) în care se inlocuiesc formlele particulare ale lui $m(t)$ corespunzând celor trei modele, presupunândcă datele de selectie sunt cenzurate *pe intervale.*

3.3 Modelul lui Musa referitor la timpul operaţional

Musa [25] a avut pentru prima dată ideea utilizării timpului operaţional ca măsură a timpului de "viaţă" al unui program. Fie τ timpul cumulat de execuţie (timpul operaţional) al unui program şi N_0 numărul iniţial de erori din soft. Fie $\mu(\tau)$ numărul de erori corectate până la momentul de timp operaţional τ. Musa presupune că funcţia rata că derilor $\lambda(\tau)$ la momentul de timp operaţional τ este proporţională cu numărul de erori rămase în soft adică este de forma

$$\lambda(\tau) = fK(N_0 - \mu(\tau)) \tag{3.13}$$

unde f şi K sunt parametri legaţi de faza de testare a programului şi anume: f este *frecvenţa liniară de execuţie.* în sensul că f reprezintă rata medie de instrucţiuni executate împărţită la numărul total de instrucţiuni ale programului. iar K este *rata de expunere la eroare* care leagă frecvenţa cu care apar erorile în raport cu frecvenţa liniară de execuţie.

Presupunând că rata cu care se corectează erorile este proporţională cu rata de apariţie a căderilor $\lambda(t)$ avem

$$\frac{d\mu(\tau)}{d\tau} = BC\lambda(\tau) \tag{3.14}$$

unde B este interpretat ca *factor de reducere a erorilor* iar C este *factor de compresie a testării.* Este normal să se introducă coeficienţi separaţi care

să caracterizeze reducerea erorilor şi procesul de testare care îşi propune să detecteze erorile.

Combinând (3.13) cu (3.14) rezultă

$$\frac{d\mu(\tau)}{d\tau} = BCfK(N_0 - \mu(\tau)). \tag{3.15}$$

Folosind pentru această ecuaţie diferenţială *condiţia iniţială naturală* $\mu(0) = 0$, soluţia cesteia devine

$$\mu(\tau) = N_0(1 - e^{-BCfK\tau}) \tag{3.15'}$$

şi ea reprezintă *funcţia lui Musa* pentru modelul bazat pe timpul de execuţie. Ea este media unui proces Poisson neomegen. Deci modelul lui Musa este un model de fiabilitate bazat pe un proces Poisson neomogen care poate fi tratat ca şi modelul GO, dar care necesită estimarea a cinci parametri a căror semnificaţie a fost evidenţiată mai sus.

Dacă în loc de numărul de erori detectate şi corectate $\mu(\tau)$ se modelează procesul căderilor utilizând numărul cumulat de căderi detectate $\nu(\tau)$ pe durata timpului de execuţie τ, se obţine un rezultat asemănător, anume

$$\nu(t) = M_0(1 - e^{-BCfK\tau}), \quad M_0 = \frac{N_0}{B}, \tag{3.15"}$$

unde B este factorul de reducere a erorilor. (Aici se vede necesitatea utilizării lui B în procesul de modelare). Modelul este de asemenea un model bazat pe PPNO de medie $\nu(\tau)$ ce foloseşte timpul de execuţie τ. Estimarea parametrilor se face asemănător modelului GO.Originalitatea modelului constă in introducerea unor coeficienţi care au interpretări *fizice* legate de particularităţile procesului de testare si de modul de manifestare a erorilor.

• **Modelul logaritmic Poisson [35].** Acesta este un model bazat pe un PPNO şi pe timp de execuţie. Aici ipoteza principală constă in aceea că intensitatea (rata) căderilor $\lambda(\tau)$ descreşte excponenţial in raport cu numărul de erori corectate $\mu(\tau,)$ adică

$$\lambda(\tau) = \lambda_0 e^{-\varphi\mu(\tau)} \tag{3.16}$$

unde λ_0 este rata căderilor iniţială iar φ este parametrul de descreştere al intensităţii de reinoire. Folosind şi relaţia dintre numărul mediu de erori

corectate şi intensitatea căderilor, adică

$$\frac{d\mu(\tau)}{d\tau} = \lambda(\tau)$$

obţinem ecuaţia diferenţială

$$\frac{d\mu(\tau)}{ds\tau} = \lambda_0 e^{-\varphi\mu(\tau)}. \tag{3.16'}$$

Ecuaţia (3.16) se mai scrie

$$\frac{d\mu(\tau)}{d\tau} e^{\varphi\mu(\tau)} = \lambda_0$$

sau

$$\frac{d}{d\tau}\left(e^{\varphi\mu(\tau)}\right) = \lambda_0.$$

Prin integrarea ultimei ecuaţii se obţine

$$e^{\varphi\mu(\tau)} = \lambda_0\tau + c, \quad c = const$$

sau

$$\mu(\tau) = \frac{1}{\varphi}\log(\lambda_0\tau + c).$$

Folosind acum condiţia iniţială naturală $\mu(0) = 0$ se obţine $c = 1$, adică ecuaţia (3.16') are soluţia

$$\mu(\tau) = \frac{1}{\varphi}\log(\lambda_0\tau + 1). \tag{3.17}$$

Din (3.16) se deduce că intensitatea căderilor este

$$\lambda(\tau) = \lambda_0 e^{-\varphi\mu(\tau)} = \frac{\lambda_0}{\lambda_0\tau + 1}, \tag{3.17'}$$

iar funcţia condiţionată de fiabilitate este

$$R(\tau|\tau_0) = e^{-\mu(\tau_0+\tau)+\mu(\tau_0)} = \frac{\lambda_0\tau_0 + 1}{\lambda_0(\tau_0 + \tau) + 1}. \tag{3.17''}$$

Modelul logaritmic Poisson este astfel rezolvat. Estimaţia parametrilor λ_0, φ se face cu ajutorul metodei verosimilităţii maxine ca în cazul oricărui model bazat pe un $PPNO$.

Pentru toate modelele bazate pe PPNO, cu ajutorul estimaţiilor $\hat{m}(t)$ se pot determina celelalte caracteristici de fiabilitate mai importante ca de exemplu $\Psi(t)$ (estimat cu $\hat{\Psi}(t) = 1/\hat{m}(t)$ şi $R(t|t_0)$, $t_0 < t$, utilizând formula (3.2).

Observaţii. Referitor la modelele bazate pe timp operaţional se pot observa următoarele:

(1). Datele asupra timpului de execuţie sunt mai concludente pentru studciul fiabilităţii programelor decât cele privind timpul calendaristic;

(2). Aceste date sunt importante pentru studiedrea *efortului de testare* a programelor, deci sunt preferate de managerii companiilor producătoare de soft.

(3). Aceste daate se obnţin însă cu destulă dificultate. De aceea Musa [25] a publicat date "istorice" ce pot fi folosite la testarea modelelor bazate pe timp operaţional.

3.4 Alte modele bazaate pe PPNO

Vom prezenta şi alte modele bazate pe PPNO, cărora le corespund diverse forme ale funcţiei $m(t)$.

• **Modelul Douane [1, 35]** sau modelul Weibull care presupune că

$$m(t) = \left(\frac{t}{\alpha}\right)^{\beta}, \alpha > 0, \beta > 0 \tag{3.18}$$

adică rata de apariţie a erorilor este

$$\lambda(t) = \frac{dm(t)}{d\lambda} = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1}. \tag{3.18'}$$

Se observă că dacă $\lambda(t)$ este rata căderilor pentru repartiţia Weibull şi dacă logaritmăm $m(t)$ avem

$$\log m(t) = \beta \log \frac{t}{\alpha} = \beta \log t - \beta \log \alpha = a + b \log t$$

$$a = -\beta \log \alpha, \quad b = \beta$$

adică o funcţie liniară. Dacă considerăm datele de observaţie $m_i = i, t_i = T_i$, (adică observaţiile naturale!), atunci paarametri a şi b se pot estima cu metoda celor mai mici pătrate din modelul liniar

$$\log m_i = a + b \log t_i + e_i, \quad (e_i = eroare).$$

Littlewood [2, 25, 35] a propus o modificare a modelului astfel

$$m(t) = k(1 - \left(\frac{\alpha}{\alpha + t}\right)^{\beta}), \alpha > 0, \beta > 0 \qquad (3.18'')$$

unde $k = N_0$ este numărul iniţial de erori ce trebuie detectate.

Se observă că

$$\lambda(t) = m'(t) = k\beta\alpha^{\beta}(\alpha + t)^{-\beta-1}$$

şi

$$\lambda(0) = \frac{k\beta}{\alpha}, \quad lim_{t\to\infty}\lambda(t) = 0.$$

Observăm că $N_0 = k$ se poate estima ca în cazul modelului J-M.

• **Modelul logistic [25]** are funcţia $m(t)$ de forma

$$m(t) = \frac{k}{1 + ae^{-bt}}, a > 0, b > 0, k > 0 \qquad (3.19)$$

care este *curba de creştere* logistică. Deoarece $m(\infty) = k$ rezultă că numărul iniţial de erori k se poate estima din modelul J-M, după care cu datele $m_i = i, t_i = T_i$ se poate liniariza relaţia (3.19), aplicând transformarea $y = \frac{k}{m(t)} - 1 = ae^{-bt}$ şi apoi logaritmând relaţia rezultată. Se obţine un model liniar în datele $y_i = \frac{1}{i} - 1, T_i$, iar cu metoda celor mai mici pătrate se estimează $\log a$ şi $-b$.

• **Modelul Gomperz [35]** este modelul pentru care $m(t)$ are forma *curbei de creştere a lui Gomperz*, adică de forma

$$m(t) = ka^{b^t}, a > 0, 1 > b > 0, k > 0. \qquad (3.20)$$

Deoarece $b < 1$ rezultă

$$m(\infty) = ka^{b^{\infty}} = ka^0 = k \qquad (3.20')$$

iar deoarece se presupune $k > 1$, rezultă că estimarea parametrilor modelului se poate face cu uşurinţă prin *metoda celor mai mici pătrate* folosind datele $m_i = i, t_i = T_i$, logaritmând in prealabil de două ori relaţia (3.20).

• **Modele PPNO bazate pe "efortul de testare" [7, 25]** presupun că media $m(t)$ a PPNO $N(t)$=numărul de erori detectate şi eliminate până la

momentul t, depinde de $w(t)$=efortul de testare (exprimat în bani) cheltuit la momentul t [25, 35] adică

$$m(t) = a(1 - e^{-rW(t)}),\ 0 < r < 1, a > 0, W(t) = \int_0^t w(u)du \qquad (3.20")$$

unde $W(t)$ este efortul de testare cumulat pe $[0, t)$. Intensitatea căderilor este în acest caz

$$\lambda(t) = \frac{dm(t)}{dt} = arw(t)e^{-rw(t)}.$$

O formă particulară a funcţiei $w(t)$ poate fi cea de tip Weibull

$$w(t) = \alpha\beta\gamma t^{\gamma-1}e^{-\beta t^{\gamma}}$$

de unde rezultă funcţia efortului de testare cumulat

$$W(t) = \int_0^t w(u)du = \alpha(1 - \beta t^{\gamma}).$$

Deci funcţia $m(t)$ a PPNO este în final

$$m(t) = a[1 - \exp\{-r\alpha(1 - e^{-\beta t^{\gamma}})\}].$$

Estimarea parametrilor acestui model se poate face fie în mod obişnuit ca la orice model bazat pe PPNO, fie estimând in prealabil a (care este numărul iniţial de erori) ca în modelul J-M, iar apoi logaritmând $m(t)$ şi aplicând metoda celor mai mici pătrate pentru a estima ceilalţi paraametri, fie aplicând direct de exemplu metoda verosimilităţii maxime.

Se poate observa că din punct de vedere matematic, modelul bazat pe efort de testare derivă din modelul G-O făcând o transformare $W(t)$ a *scalei* de variaţie a timpului. In acest model, de fapt, timpul înseamnă bani!

In [20] se propune o tratare *unitară* a modelelor de fiabilitatea programelor.

• **Modele definite prin rata periculozităţii.** Aceste modele [2] sunt caracterizate de rata periculozităţii PPNO, $\lambda(t) = \frac{dm(t)}{dt}$.

Model cu funcţia de periculozitate exponenţială liniară. Se presupune că

$$\lambda(t) = e^{\alpha_0 + \alpha_1 t},$$

iar media PPNO este in acest caz

$$m(t) = \int_0^t \lambda(u)du = \frac{e^{\alpha_0}}{\alpha_1}(e^{\alpha_1 t} - 1).$$

Deoarece $\alpha_1 < 0$ acest model coincide cu modelul Goel-Okumoto (formula (3.9)). Rezolvarea modelului (estimarea parametrilor α_0, α_1) se poate realiza prin metoda verosimilităţii maxime ca in & 3.1, sau cu metoda celor mai mici pătrate, după cum urmează. Dacă datele de selecţie sunt (i, T_i), $1 \leq i \leq n, , T_i$ = momentul de timp operaţional când are loc căderea a $i-a$, atunci parametri menţionaţi se determină rezolvând sistemul de ecuaţii neliniare

$$\frac{\partial S}{\partial \alpha_0} = 0, \ \frac{\partial S}{\partial \alpha_1} = 0$$

unde

$$S = S(\alpha_0, \alpha_1; T_1, ..., T_n) = \sum_{i=1}^{n} \left[i - \frac{e^{\alpha_0}}{\alpha_1}(e^{\alpha_1 T_i} - 1) \right]^2 .$$

Soluţia $\hat{\alpha}_0, \hat{\alpha}_1$ se obţine rezolvând numeric sistemul precedent.

Modelul exponenţial polinomial. In acest caz se presupune că

$$\lambda(t) = e^{\alpha_0 + \alpha_1 t + ... + \alpha_k t^k}, \ t > 0. \ \alpha_k < 0.$$

Şi aici se poate proceda ca în modelul precedent şi anume: se consideră

$$m(t) = \int_0^t e^{\alpha_0 + \alpha_1 u + ... + \alpha_k u^k} du,$$

se determină suma pătratelor erorilor

$$S = \sum [i - m(T_i)]^2$$

şi apoi egalând cu zero derivatele parţiale ale ale lui S în raport cu parametri necunoscuţi $\alpha_0, \alpha_1, ..., \alpha_k$, se obţine un sistem neliniar în aceşti parametri şi care se rezolvă tot cu metode numerice.

Un model combinat. Acest model presupune că

$$\lambda(t) = \beta t^{\beta - 1} e^{\alpha_0 + \alpha_1 t}, \ \beta > 0, \alpha_0, \alpha_1 \in \mathbf{R}.$$

Dacă $\beta = 1$ atunci modelul se reduce la modelul exponenţial liniar iar dacă $\alpha_1 = 0$, atunci acest model se reduce la modelul Douane, tratat la inceputul acestei secţiuni.

După cum am constatat cu ocazia modelului logaritmic Poisson, funcţia $m(t)$ poate fi dedusă dintr-o ecuaţie diferenţială care exprimă anumite proprietăţi sau condiţii satisfăcute de această funcţie. In cele ce urmează vom considera şi alte modele bazate pe astfel de ecuaţii.

3.5 Modele bazate pe ecuaţii diferenţiale

Vom folosi in continuare (vezi [2, 25, 30]) unele proprietăţi ale funcţiei *Cummulative Mean Time Between Failures,* $CMTBF(t) = \Psi(t)$ care satisface (3.2'), unde $m(t) = E[N(t)]$ este media PPNO. Funcţia $\Psi(t) = 1/m(t)$ are unele proprietăţi *naturale* ce vor fi introduse şi utilizate mai jos.

• **Modelul cu termen exponenţial simplu** presupune în mod natural că

$$lim_{t\to\infty}\Psi(t) = K_3, \frac{d\Psi(t)}{dt} = K_2(K_3 - \Psi(t)) \tag{3.21}$$

iar rezolvând ecuaţia diferenţială precedentă obţinem

$$\Psi(t) = K_3(1 - K_1 e^{-K_2 t}). \tag{3.21'}$$

Se observă că K_3 este numărul iniţial de erori. Estimarea parametrilor se poate face ca în cazul oricărui model bazat pe PPNO ţinând seama că $m(t) = 1/\Psi(t)$, sau aplicând relaţiei (3.21') metoda celor mai mici pătrate (în cazul neliniar!), cu datele

$$\Psi_i = \frac{T_i}{i}, \; t_i = T_i.$$

Prima formulă (3.21) este o ipoteză *plauzibilă* pe care se bazează modelul.

•**Modelul lui Lloyd-Lipow** [2] presupune că rata de variaţie a lui $\Psi(t)$ este invers proporţional cu pătratul timpului adică

$$\frac{d\Psi(t)}{dt} = \frac{K_2}{t^2} \tag{3.22}$$

de unde rezolvând ecuaţia diferenţială avem

$$\Psi(t) = \begin{cases} K_3 - \frac{K_2}{t} & \text{dacă} \quad t \geq \frac{K_2}{K_3} \\ 0 & \text{altfel.} \end{cases} \tag{3.22'}$$

Aici de asemenea $K_3 = lim_{t\to\infty}\Psi(t)$ iar K_2 defineşte intervalul minimal $[0, K_2/K_3]$ pentru care nu există căderi. Modelul nu este întotdeauna aplicabil. In schimb parametrii săi pot fi cu uşurinţă estimaţi prin metoda celor mai mici pătrate care conduce la rezolvarea unui sistem liniar în K_2, K_3.

• **Un nou model** (vezi [30]) presupune că rata de variaaţie a funcţiei $\Psi(t)$ satisface proprietatea

$$\frac{d\Psi}{dt} = K_2 te^{-t}(\Psi(t) + K_1),\ K_1, K_2 > 0 \tag{3.23}$$

Soluţia ecuaţiei, cu condiţia iniţială $\Psi(0) = 0$, este

$$\Psi(t) = K_1(e^{-K_2e^{-t}(t+1)+K_2} - 1) \tag{3.23'}$$

şi ea satisface condiţia

$$K_3 = \lim_{t\to\infty}\Psi(t) = K_1(e^{K_2} - 1). \tag{3.23"}$$

Estimarea parametrilor din (3.22') se poate realiza cu metoda celor mai mici pătrate aplicată selecţiei bidimensionale $(\Psi_i = T_i/i, T_i), 1 \leq i \leq n$. Se obţine un sistem de ecuaţii în parametrii K_1, K_2 care nu este liniar, dar care poate fi tratat numeric.

O altă versiune a acestui model se obţine dacă rata de variaţie a funcţiei $\Psi(t)$ satisface relaţia

$$\frac{d\Psi}{dt} = K_2e^{-t}(\Psi(t) + K_1),\ K_1, K_2 > 0. \tag{3.24}$$

In mod analog, soluţia ecuaţiei (3.24) este

$$\Psi(t) = K_1(e^{K_2(1-e^{-t})} - 1) \tag{3.24'}$$

iar

$$\lim_{t\to\infty}\Psi(t) = K_3 = K_1(e^{K_2} - 1). \tag{3.24"}$$

Şi aici estimarea parametrilor K_1, K_2 se realizează cu metoda celor mai mici pătrate, care conduce tot la un sistem de ecuaţii neliniare.

• **Modelul lui Roesner** [2, 30] este unul din primele modele bazate pe ecuaţii diferenţiele. Altfel, dacă se notează cu $D(t)$ numărul mediu de

erori rămase în sistem şi se presupune că fiecare cădere este determinată de o eroare care este detectată odată cu apariţia căderii şi este eliminată, modelul presupune că rata de variaţie a lui $D(t)$ satisface relaţia

$$\frac{dD(t)}{dt} = -K_2 D(t),\ t > 0,\ K_2 > 0,\ D(0) = K_1. \tag{3.25}$$

Deci K_1 este *numărul iniţial mediu de erori* (adică N_0 din modelul J-M!).

Soluţia ecuaţiei (3.25) este

$$D(t) = K_1 e^{-K_2 t} \tag{3.25'}$$

de unde se deduce că numărul mediu de erori eliminate pe intervalul $[0, t]$ este

$$m(t) = K_1(1 - e^{-K_2 t}) \tag{3,25"}$$

care coincide cu modelul Goel-Ocumoto.

Dacă se cunoaşte K_2 (estimat cu modelul GO) atunci $\xi = 1 - e^{-K_2 t}$ este proporţia erorilor eliminate până la momentul t. Presupunând că efortul de testare pentru a se elimina proporţia ξ de erori a costat C unităţi băneşti atunci costul necesar pentru a elimina încă proporţia ξ_1 de erori este, conform unui calcul liniar

$$C_1 = \frac{C\xi_1}{\xi} \tag{3.26}$$

formulă de calcul simplă dar şi utilă pentru producătorii de soft.

2.6 Modele pentru mai multe tipuri de erori

Una din criticile modelelor de fiabilitatea programelor constă în aceea că se presupune că toate erorile au aceeaşi mărime şi importanţă şi că erorile se pot manifesta cu aceeaşi intensitate. O slăbire a acestei presupuneri constă în a considera că erorile sunt de două tipuri [25, 35], adică media PPNO este

$$m(t) = a_1(1 - e^{-b_1 t}) + a_2(1 - e^{-b_2 t}) \tag{3.27}$$

unde a_1 este numărul iniţial de erori de tipul 1 , a_2 este numărul iniţial de erori de tipul 2, b_1 este rata de detectare a erorilor de tipul 1 iar b_2 este rata de detectare a erorilor de tipul 2. Notând $a = a_1 + a_2$ numărul total de erori avem

$$p_i = \frac{a_i}{a},\ i = 1, 2 \tag{3.27'}$$

unde p_i este probabilitatea apariţiei în execuţia programului a unei erori de tipul i. Intensitatea de apariţie a erorilor este

$$\lambda(t) = \sum_{i=1}^{2} a_i b_i e^{-b_i t} = a. \sum_{i=1}^{2} p_i b_i e^{-b_i t}. \qquad (3.27")$$

Se observă că PPNO care stă la baza acestui model este o compunere de două PPNO.

Acest model este o generalizare a modelului GO, iar estimarea parametrilor a_1, a_2, b_1, b_2 se realizează în mod asemănător modelului GO. Modelul bazat pe (3.27) poate fi încă extins la k tipuri de erori, în acest caz fiind necesară estimarea a $2k$ paarametri, iar PPNO este în acest caz rezultatul compunerii a k PPNO.

Un model simplificat *cu k tipuri de erori* se bazează pe funcţia de tip Erlang

$$m(t) = a\left[1 - e^{-bt} \sum_{i=1}^{k-1} \frac{(bt)^j}{j!}\right] \qquad (3.29)$$

unde a este numărul iniţial de erori, b este rata de detectare a erorii (independenta de tipul erorii, când procesul se stabilizează) iar k este un parametru intreg. Dacă $k = 1$ acest model se reduce la modelul GO iar dacă $k = 2$ modelul este "in formă de S" (de tipul (3.10)). Altfel, modelul se bazează pe un PPNO cu trei parametri a, b, k ce trebuie estimaţi prin metodele obişnuite aplicate celorlalte modele de acest fel.

4 Modele privind optimizarea realizării programelor

Printre problemele legate de dezvoltarea şi realizarea produselor soft (probleme de ingineria de software şi managementul acestei activităţi) sunt şi cele legate de optimizarea activităţilor, evaluarea şi optimizarea costurilor activităţilor de testare şi intrteţinere, precum şi cele privind selectarea variantelor sau versiunilor convenabile sau optime. Referitor la astfel de probleme s-au scris multe lucrări, s-au construit diverse modele, s-au elaborat chiar produse software [1, 18]. In cele ce urmează vom prezenta unele din aceste modele, cele mai reprezentative.

4.1 Modele pentru determinarea duratei de testare

Se ştie că testarea produselor soft este o operaţie migăloasă, costisitoare şi indelungată. In aceste condiţii producătorii de software sunt interesaţi să poată decide când este momentul convenabil să oprească activitatea de testare şi să dea produsul în exploatare. Acest moment trebuie ales astfel încât pe de-o parte să se asigure o fiabilitate mare programului prin eliminarea unui cât mai mare număr de erori, iar pe de altă parte costurile operaţiilor de realizare să fie cat mai mici, sau convenabil de mici. Criteriul de optim considerat în aceste modele constă în *minimizarea costurilor.* Operaţia de testare se termină când s-a eliminat un număr de erori sau când s-a parcurs o anumită durată, ambele determinate din condiţia de cost minim. Astfel de modele sunt numite în literatura de specialitate [18, 23] *modele de creştere a fiabilităţii programelor.*

Pentru modelele ce for fi prezentate în această secţiune se presupun satisfăcute următoarele ipoteze:

1^0. Orice cădere a programului este determinată de o singură eroare existentă in program;

2^0. Corectarea unei erori se face instantaneu (timpul de corectare a erorii este neglijabil);

3^0. Prin eliminarea erorii nu se mai introduc alte erori.

Strategia optimă de testare va consta in determinarea numărului optim de erori corectate după testarea programului. Vom presupune că durata de funcţionare fără căderi a programului are o repartiţie de probabilitate DFR,

(sau DFI). Vom folosi de asemenea notaţiile:

n– numărul de erori detectate până la terminarea testării;

n^*– valoarea optimă a lui n;

T_k– momentul detectării erorii k;

C_1– costul unei unităţi de timp de testare;

C_2– costul corectării unei erori în cursul testării;

C_3– costul corectării unei erori în timpul fazei operaţionale (adică după ce programul este dat în exploatare); $C_3 > C_2$.

Câştigul mediu este

$$G(n) = (C_3 - C_2).n - C_1.E\{T_n\}. \tag{4.1}$$

Observaţia 4.1. *Dacă repartiţia căderilor este DFR atunci* $E(T_{n+1} - T_n)$ *este crescătoare în* n.

Demonstraţie. Se observă că

$$E(T_{n+1} - T_n) = E\{E(T_{n+1} - t_n | T_n = t_n)\} = E(L(t_n))$$

unde t_n este valoarea observată a lui T_n şi $L(t)$ este media timpului de funcţionare rămas până la defectare, adică

$$L(t) = E(T - t | T > t) = \frac{1}{\overline{F}(t)} \int_t^\infty \overline{F}(x)dx. \tag{4.2}$$

Din (4.2) rezultă că

$$\overline{F}(t) = exp\left(-\int_0^t \frac{1 + L'(x)}{L(x)} dx\right).$$

Dar F este DFR conduce la $L(t_n)$-crescătoare în t_n, deci $t_n > t_{n-1} \Rightarrow L(t_n) > L(t_{n-1})$ adică $E(L(t_n)) > E(L(t_{n-1}))$ care este crescătoare în n.

Suntem acum in măsură să enunţăm următoarea

Teorema 4.1. *Dacă repartiţia căderilor este DFR atunci funcţia de câştig mediu (4.1) are o soluţie optimă unică.*

Demonstraţie. Condiţia necesară ca o valoare n să fie punct de maxim pentru $G(n)$ este

$$G(n-1) < G(n) \quad si \quad G(n) \geq G(n+1), \tag{4.3}$$

sau echivalent

$$E(T_{n+1} - T_n) \geq k \quad si \quad E(T_n - T_{n-1}) < k \tag{4.3'}$$

unde

$$k = \frac{C_3 - C_2}{C_1}.$$

Să analizăm câteva cazuri:

- dacă $E(T_1) \geq k$, atunci $n^* = 0$, adică programul se dă în exploatare fără testare;

- dacă $k \geq \lim_{n\to\infty} E(T_n - T_{n-1})$, atunci programul se dă în exploatare numai după ce s-au eliminat toate erorile;

- în celălălalt caz (cazul interesant) soluţia optimă n^* se obţine din condiţia (4.3) sau (4.3'). Teorema este demonstrată. De aici se poate construi cu uşurinţă un algoritm pentru determinarea lui n^*.

4.1.1. Model bazat pe utilizarea efortului de testare.

Să considerăm acum modelul Poissonian al lui Yamada et al. [23,36] în care funcţia $m(t)$, numărul mediu de erori sau media procesului Poisson, este dată de ecuaţia diferenţială

$$\frac{dm(t)}{dt} \cdot \frac{1}{w(t)} = \lambda(N_0 - m(t)), \tag{4.4}$$

unde $w(t)$ este *funcţia de cost a efortului de testare la momentul* t iar N_0 este numărul iniţial de erori din soft. Din (4.4) rezultă că $m(t)$ este dat de relaţia

$$m(t) = N_0(1 - exp(-\lambda.W(t))), \quad W(t) = \int_0^t w(x)dx \tag{4.5}$$

iar $W(t)$ se mai poate scrie

$$W(t) = \frac{1}{\lambda} \ln\left(\frac{N_0}{N_0 - m(t)}\right). \tag{4.5'}$$

Fiind dată funcţia efort de testare $w(t)$ (sau varianta sa cumulată $W(t)$) să determinăm [23] *durata optimă de testare* a programului. Vom folosi notaţiile:

T_{LC} – lungimea ciclului de viaţă a programului, măsurată din momentul inceperii testării;

$T-$ timpul total de testare, până când programul intră in exploatare;
T^*- valoarea optimă a lui T.
Atunci, costul mediu total se scrie

$$C(T) = C_2.m(t) + C_3(m(T_{LC}) - m(T)) + C_1 \int_0^T W(x)dx. \qquad (4.6)$$

Condiţia de cost minim este

$$\frac{\partial C(T)}{\partial T} = 0 \Rightarrow A(T) = \frac{C-1}{C_3 - C_2}, \quad A(T) = \frac{dm(t)}{dt}.\frac{1}{w(t)}. \qquad (4.7)$$

Suntem acum în măsură să formulăm teorema

Teorema 4.2. *Valoarea otpimă T^* este determinată astfel:*

a) dacă $A(0) \leq \frac{C_1}{C_3-C_2}$ atunci $T^* = 0$;

b) dacă $A(0) > \frac{C_1}{C_3-C_2} > A(T_{LC})$ atunci (4.7) are 0 soluţie unică T_0 şi $T^* = T_0$;

c) dacă $A(T_{LC}) \geq \frac{C_1}{C_3-C_2}$ atunci $T^* = T_{LC}$.

Demonstraţie. Fie T^* valoarea lui T care minimizează (4.6). Observând că $A(T)$ este monoton descrescătoare, vom analiza pe rând cele trei cazuri.

a) Dacă $A(0) \leq 0$ atunci trebuie să avem $A(T) < \frac{C_1}{C_3-C_2}, 0 < T < T_{LC}$, deci $\frac{\partial C}{\partial T} > 0, 0 < T < T_{LC}$; rezultă că $T^* = 0$.

b) Dacă $A(0) > \frac{C_1}{C_3-C_2} > A(T_{LC})$ atunci (4.7) are o soluţie unică T_0 care satisface condiţiile

$$A(T) > \frac{C_1}{C_3 - C_2}. \quad cand \quad 0 < T < T_0:$$

$$A(T) < \frac{C_1}{C_3 - C_2}. \quad cand \quad T_0 < T < T_{LC}$$

Deoarece $\frac{\partial C}{\partial T} < 0, 0 < T < T_0$ şi $\frac{\partial C}{\partial T} > 0, T_0 < T < T_{LC}$ rezultă că $T^* = T_0$.

c) Dacă $A(T_{LC}) \geq \frac{C_1}{C_3-C_2}$ atunci $A(T) > \frac{C_1}{C_3-C_2}, 0 < T < T_{LC}$ şi deoarece $\frac{\partial C}{\partial T} < 0, 0 < T < T_{LC}$ rezultă $T^* = T_{LC}$.

Demonstraţia este completă.

4.1.2 Model pentru software cu structură modulară.

Vom prezenta în continuare un model pentru timpul optim de testare preluat din [23].

Presupunem că un produs soft constă din k module interconectate. In perioada de testare T sistemul soft funcţionează efectiv timpul $\alpha_T T$, $0 < \alpha_T \geq 1$, iar în perioada de exploatare, parametrul corespunzător este α_U. Se presupune de asemenea că modulele sunt apelate pe perioada de exploatare a sistemului în proporţii diferite $\beta_j, 1 \leq j \leq k$ cu

$$\sum_{j=1}^{k} \beta_j = 1, \quad ,0 < \beta_j < 1,$$

iar ponderile β_j rămân constante pe timpul exploatării programului. Deci timpul consumat de modulul j in faza de testare este $\beta_j \alpha_T T$.

Fie N_j numărul de erori din modulul j şi presupunem că durata cumulată de funcţionare a modulului j până la detectarea unei erori este o variabilă aleatoare X_j cu funcţia de repartiţie $F_j(x; \theta_{j1}, \theta_{j2}, ..., \theta_{jm_j})$, unde θ_j este un vector de parametri statistici necunoscuţi; variabilele $X_j, 1 \leq j \leq k$ sunt presupuse independente două câte două. Presupunem satisfăcute ipotezele $2^0, 3^0$ precizate la începutul acestei secţiuni. Presupunem ca deobicei că funcţiile de repartiţie F_j sunt de tipul DFR. Notăm $t_j(i), 1 \leq i \leq N_j$ momentele de când se detectează erori in modulul j, pe parcursul perioadei de testare. Desigur $0 \leq t_j(1) \leq t_j(2) \leq ... \leq t_j(N_j) \leq S$, unde $S = \beta_j \alpha_T . T$. Deci valorile de selecţie $t_j(i)$ sunt cenzurate la dreapta de S (sunt date cenzurate de tipul I).

Parametri $\theta_{j1}, \theta_{j2}, ..., \theta_{jm_j}$ se vor estima folosind metoda verosimilităţii maxime. Funcţia de verosimilitate este

$$L(t_j(.); \theta_{j1}, \theta_{j2}, ..., \theta_{jm_j}) = \prod_{i=1}^{N_j} F_j(t_j(i); \theta_{j1}, \theta_{j2}, ..., \theta_{jm_j}).$$

Din condiţia de verosimilitate maximă se obţine sistemul

$$\frac{\partial ln L(t_j(.); \theta_{j1}, \theta_{j2}, ..., \theta_{jm_j})}{\partial \theta_{jl}} = 0, \quad 1 \leq l \leq m_j, \tag{4.8}$$

din care prin rezolvare se obţin estimaţiile $\hat{\theta}_{j1}, \hat{\theta}_{j2}, ..., \hat{\theta}_{jm_j}, 1 \leq j \leq k$.

Dacă la sistemul de ecuaţii (4.8) adăugăm şi ecuaţiile

$$\frac{\partial ln L}{\partial N_j} = 0, 1 \leq j \leq k$$

atunci se pot obţine şi estimaţii $\hat{N}_j$ pemtru N_j.

In [23] se dă un model şi algoritmul corespunzător pentru determinarea perioadei optime de testare T^*. Se presupune că se pleacă de la o perioadă iniţială dată de lungime T şi pentru a se decide dacă testarea trebuie oprită sau continuată se calculează următoarea funcţie obiectiv, care este de tip *profit*:

$$V(T) = V_1(T+\tau) - V_2(\tau) - V_3(T+\tau), \tag{4.9}$$

unde

$V_1(t)$– este *valoarea* sistemului soft la momentul t;

$V_2(t)$– este costul mediu datorat nedetectării erorilor în perioada de testare (el poate cumula şi eventuale penalizări);

$V_3(t)$– este costul cumulat al testării sistemului până la momentul t, când ar urma să intre in exploatare.

Funcţiile V_i propuse în [23] sunt următoarele:

a) Funcţia $V_1(t)$ se presupune descrescătoare in t de forma

$$V_1(t) = \lambda_1.e^{-\lambda_2 t}.H(t_D - t) + \mu_1.e^{-\mu_2 t}.(1 - H(t_D - t)) \tag{4.10}$$

unde t_D este momentul când sistemul devine operaţional, $\lambda_i, \mu_i, i = 1,2$ sunt constante pozitive, iar $H(x)$ este funcţia treaptă a lui Heaviside, adică

$$H(x) = \begin{cases} 0, & cand \quad x < 0 \\ 1, & cand \quad x \geq 0 \end{cases}.$$

b) Funcţia cost mediu V_2 este de forma

$$V_2(\tau) = \sum_{j=1}^{k} c_j(\hat{N}_j - r_j).\frac{\hat{F}_j(\beta_j(\alpha_T(T+\tau) + \alpha_U.W)) - \hat{F}_j(\beta_j\alpha_T(T+\tau))}{1 - \hat{F}_j(\beta_j\alpha T)} \tag{4.11}$$

unde o eroare care este detectată în perioada operaţională (de exploatare) W are costul c_j pentru modulul j.

c) Funcţia V_3 se ia liniară de forma

$$V_3(t) = \gamma t, \quad \gamma > 0. \tag{4.12}$$

Cu această alegere a funcţiilor de cost, algoritmul de determinare a perioadei optime de testare este

1. Intrare parametri $\beta_j, 1 \leq j \leq k, \alpha_T, \alpha_U$; intrare T =durata iniţială de testare şi W ciclul de viaţă al sistemului.

2. Intrare datele de selecţie $t_j(i), 1 \leq i \leq r_j, 1 \leq j \leq k$.

   repeat

3. Determină estimaţiile $\hat{\theta}_{j1}, \hat{\theta}_{j2}, ..., \hat{\theta}_{jm_j}, \hat{N}_j, 1 \leq j \leq k$.

4. Determină profitul $V(0)$ conform formulei (4.9). (Acest pas se execută numai la prima trecere prin ciclul **repeat**).

5. Pentru o valoare mică dată Δ (care este τ din (4.11)), se estimează $V(\Delta)$ care reprezintă profitul ce s-ar realiza dacă perioada de testare s-ar prelungi cu Δ, adică T devine $T + \Delta$.

6. Dacă $V(0) < V(\Delta)$ atunci ia $T := T + \Delta$ şi adaugă noi date de selecţie $t_j(i)$ pentru intervalul de timp $[T, T + \Delta]$. Ia $V(0) := V(\Delta)$.

   until $V(0) \geq V(\Delta)$.

7. Stop.

La terminarea algoritmului valoarea lui T este cea optimă.

Desigur, pentru aplicarea modelului trebuie să se estimeze î n prealabil parametrii $\lambda_i, \mu_i, i = 1, 2, c_j, 1 \leq j \leq k, \gamma$ (adică să se realizeze *specificarea* modelului). Acest lucru se realizează ca deobicei folosind date istorice.

4.2 Modele de alegere optimă a produselor software

Modelele care urmează îşi propun [23] să maximizeze fiabilitatea produselor soft cu cheltuieli limitate la o sumă disponibilă.

4.2.1. Modele referitoare la programe ce realizează anumite funcţii date.

Pentru aceste modele vom face următoarele ipoteze:

1*. Se cunosc fiabilitatea şi costul fiecărui program;

2*. Utilizarorul dispune de o sumă B de bani dată pentru achiziţionarea produsului software;

3*. Se cunoaşte frecvenţa de utilizare a fiecărei funcţii realizată de produsul software.

Vom folosi următoarele notaţii:

K = numărul de funcţii pe care trebuie să le realizeze produsul program;

F_k = frecvenţa de utilizare a produsului k. $1 \leq k \leq K$;
m_k = numărul de programe disponibile pentru funcţia k;
R_{kj} = fiabilitatea programului j când acesta execută funcţia k;
C_{kj} = costul programului j când execută funcţia k;
$\overline{B}$= fiabilitatea medie a produsului software;

$$X_{kj} = \begin{cases} 1, & \textit{daca programul } j \textit{ ar fi ales pentru functia } k \\ 0, & \textit{in rest.} \end{cases}$$

*** Model de alegere optimă a programelor fără redundanţă.** Variabilele X_{kj} sunt variabile de decizie şi ele trebuie determinate astfel încât să conducă la alegerea unei mulţimi de programe *fără redundanţă* din condiţia:

$$\max(\overline{R} = \sum_{k=1}^{K} F_k R_k) \tag{4.13}$$

unde

$$R_k = \sum_{j=1}^{m_k} X_{kj} R_{kj}$$

cu restricţiile

(i) $\sum_{j=1}^{m_k} X_{kj} = 1, \quad 1 \leq k \leq K;$

(ii) $\sum_{k=1}^{K} \sum_{j=1}^{m_k} X_{kj} C_{kj} \leq B, \quad X_{kj} \in \{0,1\}, \quad 1 \leq k \leq K, 1 \leq j \leq m_k.$

(Alegerea programelor *fără redundanţă* înseamnă că o funcţie va fi implementată într-un singur program achiziţionat conform restricţiei (i); alegerea *cu redundanţă* inseamnă că funcţiile pot fi realizate şi de mai multe programe).

Maximizarea funcţiei obiectiv de forma (4.13) cu restricţiile (i) şi (ii) este o problmă de programare liniară discretă în care X_{kj} sunt necunoscute. Din formularea problemei rezultă că cele K programe realizează funcţii independente. Restricţia (i) exprimă alegerea mulţimii programelor fără redundanţă.

Maximizarea fiabilităţii medii $\overline{R}$ este echivalentă cu minimizarea ratei medii de defectare a programelor dată de expresia

$$\overline{z} = \frac{N}{H} \sum_{k=1}^{K} F_k (1 - R_k), \tag{4.14}$$

unde H = timpul operaţional şi N = numărul de execuţii in perioada H. Intr-adevăr, deoarece ponderile de utilizare satisfac relaţia

$$\sum_{k=1}^{K} F_k = 1$$

rezultă

$$\overline{z} = \frac{N}{H}(1 - \overline{R}). \tag{4.14'}$$

Rezolvarea problemei de programare liniară cu variabile $\{0,1\}$ se poate face cu algoritmi cunoscuţi sau cu produse soft specializate.

** **Model de alegere optimă a programelor cu redundanţă.** In acest caz variabilele $X_{kj}, 1 \leq k \leq K, 1 \leq j \leq m_k$ se determină din relaţia

$$max\overline{R} = \sum_{k=1}^{K} F_k R_k, \tag{4.15}$$

unde

$$R_k = 1 - \prod_{j=1}^{m_k}(1 - R_{kj}) \tag{4.15'}$$

cu restricţiile

$$(i^*) \sum_{j=1}^{m_k} X_{kj} \geq 1, \quad 1 \leq k \leq k;$$

$$(ii^*) \sum_{k=1}^{K} \sum_{j=1}^{m_k} X_{kj} C + kj \leq B, \quad X_{kj} \in \{0,1\}, \quad 1 \leq k \leq K, 1 \leq j \leq m_k.$$

Condiţia de redundanţă este dată de restricţia (i^*) care spune că se va selecta cel puţin un program pentru fiecare funcţie. Fiabilitatea R_k a funcţiei k este probabilitatea ca cel puţin unul dintre programele selectate pentru funcţia k să funcţioneze (cazul sistemului paralel).

Funcţia obiectiv (4.15) poate fi rescrisă astfel:

$$\overline{R} = \sum_{k=1}^{K} F_k R_k = \sum_{k=1}^{K} F_k \left(1 - \prod_{j=1}^{m_k} F_k (1 - R_{kj})^{X_{kj}}\right)$$

$$= 1 - \sum_{k=1}^{K} F_k \prod_{j=1}^{m_k} (1 - R_{kj})^{X_{kj}}. \tag{4.16}$$

Din (4.16) rezultă că problema maximizării funcţiei $\overline{R}$ este echivalentă cu problema de minimizare

$$min \sum_{k=1}^{K} F_k \prod_{j=1}^{m_k} (1 - R_{kj})^{X_{kj}}. \tag{4.16'}$$

Problema de optimizare (4.15) sau (4.16') este o problemă de programare dinamică discretă. Soluţia modelului ** cu funcţia obiectiv (4.16) se obţine cu ajutorul unui algoritm ce va fi descris mai jos. Să notăm $R_k(S)$ fiabilitatea sistemului software care satisface funcţiile $k, k+1, ..., K$ când suma de bani alocată pentru procurarea lor este S (k este interpretat ca *starea* curentă in algoritmul de programare dinamică, retrospectiv).

Algoritmul **

1. Pentru starea finală K se calculează

$$R_(S) = \min F_K \prod_{j=1}^{m_K} (1 - R_{Kj})^{X_{Kj}}, \tag{4.17}$$

cu restricţiile

$$\sum_{j=1}^{m_K} X_{Kj} \geq 1, \quad \sum_{j=1}^{m_K} C_{Kj} X_{Kj} \leq S, \tag{4.18}$$

unde

$$S \in \left\{ \min_j C_{Kj}, B - \sum_{i=1}^{K-1} \min_j C_{ij} \right\}. \tag{4.19}$$

2. Pentru $k = K-1, K-2, ..., 2, 1$ se calculează

$$R_k(S) = min \left\{ F_k \sum_{j=1}^{m_k} (1 - R_{kj})^{X_{kj}} + R_{k+1}(S - \sum_{j=1}^{m_k} C_{kj} X_{kj}) \right\}, \tag{4.20}$$

cu restricţiile

$$\sum_{j=1}^{m_k} X_{kj} \geq 1, \quad \sum_{j=1}^{m_k} C_{kj} X_{kj} \leq S, \tag{4.21}$$

unde

$$S \in \left\{ \sum_{i=k}^{K} \min_j C_{ij}, B - \sum_{i=1}^{k-1} C_{ij} \right\}. \tag{4.22}$$

(Se observă că (4.17) şi (4.22) asigură selectarea a cel puţin un program pentru fiecare funcţie).

3. Se determină fiabilitatea medie maximă

$$\overline{R^*} = 1 - \min_S R_1(S), \tag{4.23}$$

şi politica optimă X^*_{kj}, $k = 1, 2, ..., K$; $j = 1, 2, ..., m_k$ ce corespunde lui $\overline{R^*}$, unde dacă $X^*_{kj} = 1$ înseamnă că s-a selectat programul j pentru funcţia k.

4. Stop.

Menţionăm din nou că valorile X^*_{kj} reprezintă soluţia optimă.

4.2.2. Modele bazate pe costuri pentru soft cu structură modulară.

Se presupune că produsul software se compune din mai multe versiuni, fiecare cu mai multe module. Vom Presupune că sunt satisfăcute ipotezele $1^*, 2^*, 3^*$ şi vom folosi de asemenea unele din notaţiile deja introduse cu următoarele completări:

n = numărul de module care compun sistemul de programe (produsul soft);

m_i = numărul de versiuni disponibile pentru modulul i;

R_{ij} = fiabilitatea (presupusă estimată) pentru versiunea j a modulului i;

C_{ij} =costul versiunii j a modulului i;

R_i = fiabilitatea modulului i;

R = fiabilitatea sistemului de programe;

$$X_{ij} = \begin{cases} 1 & daca\ pentru\ modulul\ i\ este\ selectata\ versiunea\ j; \\ 0, & in\ rest. \end{cases}$$

Variabilele X_{ij} sunt variabile de decizie: ele indică veriunea şi modulul selectat din fiecare versiune.

• **Model pentru selectarea unei mulţimi optime de module cu o singură funcţie. Versiunea fără redundanţă.** Acest model foloseşte funcţia obiectiv

$$\max(R = \prod_{i=1}^{n} R_i), \tag{4.24}$$

unde

$$R_i = \sum_{j=1}^{m_i} X_{ij} R_{ij},$$

cu restricţiile

$(i^{\spadesuit})$ $\sum_{j=1}^{m_i} X_{ij} = 1, \quad i = 1,2,...,n;$

$(ii^{\spadesuit})$ $\sum_{i=1}^{n}\sum_{j=1}^{m_i} X_{ij}C_{ij} \leq B, \quad X_{ij} \in \{o,1\}, \quad 1 \leq i \leq n, \quad 1 \leq j \leq m_i.$

Funcţia obiectiv (4.24) presupune alegerea unei singure versiuni pentru fiecare modul, din care cauză soluţia selectată se va nota $X_1, X_2, ..., X_n$ unde $X_i = 1$ înseamnă $X_{ij_i} = 1$.

Soluţia modelului se obţine printr-un algoritm *branch and bound* [24].

Pentru construirea algoritmului să observăm următoarele:

1^0 Problema (4.24) are soluţie dacă

$$\sum_{i=1}^{n}\left(\min_j C_{ij}\right) \leq B. \tag{4.25}$$

2^0 Dacă (4.25) are loc, atunci o soluţie posibilă a problemei este dată de alegerea celei mai ieftine versiuni pentru fiecare modul (acesta putând reprezenta o margine inferioară).

3^0 Dacă se omite restricţia $(ii^{\spadesuit})$ atunci o soluţie optimă a problemei este $(X_1^0, X_2^0, ..., X_n^0)$ cu valoarea funcţiei obiectiv

$$\prod(\max_i R_{ij}). \tag{4.26}$$

Algoritmul constă din următorii paşi [23]:

1. Se alege o soluţie posibilă a problemei şi se calculează marginea inferioară M_{inf} a funcţiei obiectiv;

2. *Nodul rădăcină* (al algoritmului "branch and bound"!), adică "nivelul 0" $(i = 0)$, este ramificat in m_1 ramuri, fiecare corespunzând unei alegeri a lui $X_{1j}, 1 \leq j \leq m_1$;

3. La nivelul curent i se alege o versiune pentru modulul $i, 1 \leq i \leq n$;

4. Pentru fiecare soluţie parţială $(X_1, X_2, ..., X_k), k < n$, se calculează:

   - marginea superioară M_{sup}, definită ca valoarea funcţiei obiectiv corespunzătoare soluţiei $(X_1, ..., X_k, X_{k+1}^0, ..., X_n^0)$

$$M_{sup} = \left(\prod_{i=1}^{k} R_i\right)\left(\prod_{i=k+1}^{n} \max_j R_{ij}\right); \tag{4.27}$$

- costul soluţiei margine superioară CM_{sup} dat de formula

$$CM_{sup} = \sum_{i=1}^{k} C_i + \sum_{i=k+1}^{n} C_i^0; \tag{4.28}$$

- costul minim posibil C_{min} dat de formula

$$C_{min} = \sum_{i=1}^{k} C_i + \sum_{i=k+1}^{n} (\min_j C_{ij}); \tag{4.29}$$

- valoarea funcţiei obiectiv corespunzătoare soluţiei care dă costul minim C_{min}

$$RC_{min} = \left(\prod_{i=1}^{k} R_i\right)\left(\prod_{i=k+1}^{n} R_{ij_c}\right), \tag{4.30}$$

unde R_{ij_c} este fiabilitatea celei mai iefrine versiuni a modulului i.

5. O soluţie parţială este eliminată dacă

$$C_{min} > B \quad sau \quad M_{sup} < M_{inf}. \tag{4.31}$$

6. Soluţiile parţiale pentru care $C_{min} = B$ nu se mai ramifică, însă valoarea RC_{min} este reţinută pentru analiza finală, ca posibilă soluţie.

7. Următorul nod de ramificate este ales nodul cu cea mai mare margine superioară, cu condiţia $CM_{sup} > B$.

8. Nodurile corespunzătoare soluţiilor parţiale pentru care $CM_{sup} \leq B$ nu se mai ramifică, însă valorile funcţiei obiectiv sunt reţinute ca posibile soluţii finale.

9. Stop.

Algoritmul se termină când s-a ales o versiune pentru fiecare modul, adică s-au explorat n nivele ale arborelui asociat algoritmului "branch and bound".

•• **Model pentru selectarea unei mulţimi optime de module cu o singură funcţie. Versiunea cu redundanţă.** Cu notaţiile intreduse, acest model se bazează pe funcţia obiectiv

$$\max\{R = \prod_{i=1}^{n} R_i\} \tag{4.32}$$

unde

$$R_i = 1 - \prod_{j=1}^{m_i}(1 - R_{ij})^{X_{ij}}$$

cu restricţiile

(i^0) $\sum_{j=1}^{m_i} X_{ij} \geq 1, \quad i - 1, 2, ..., n$

(ii^0) $\sum_{i=1}^{n}\sum_{j=1}^{m_i} X_{ij}C_{ij} \leq B, \quad X_{ij} \in \{0,1\}, \quad i = 1,2,...,n \quad j = 1,2,...,m_i.$

Modelul se rezolvă cu un algoritm de programare dinamică discretă asemănăto algoritmului **.

••• Model pentru selectarea unei mulţimi de module ce pot satisface K funcţii. Versiunea fără redundanţă. Vom presupune verificate ipotezele **1***, **2***, **3*** precum şi următoarele ipoteze

4* Execuţia funcţiei k a sistemului de programe ce trebuie selectate, $k = 1,2,...,K$, presupune execuţia mulţimii de module S_k, (adică pentru orice modul $i \in S_k$ există m_i versiuni disponibile).

5* Acelaş modul conţine funcţii diferite ale sistemului soft, adică

$$S_{k1} \bigcap S_{k2} \neq \emptyset, \quad \forall k1, k2 \in \{1,2,...,K\}$$

Presupunem că toate modulele apelate de toate funcţiile sunt notate cu $1,2,...,n$, deci

$$n \leq \sum_{i=1}^{K} card(S_k).$$

Modelul are ca obiectiv

$$\max(R = \sum_{i=1}^{K} F_k \prod_{i \in S_k} R_i). \tag{4.33}$$

cu restricţiile

(i) $\sum_{i=1}^{m_i} X_{ij}, \quad i = 1,2,...,n$

(ii) $\sum_{i=1}^{n}\sum_{j=1}^{m_i} X_{ij}C_{ij} \leq B, \quad X_{ij} \in \{0,1\}, \quad i = 1,2,...,n, j = 1,2,...,m_i.$

Soluţia acestui model se poate obţine cu un algoritm de tip *"branch and bound"* discutat cu ocazia modelului definit de funcţia obiectiv (4.24) (modelul •).

••••Model pentru selectarea unei mulţimi de module ce pot satisface mai multe funcţii. Versiunea cu redundanţă. Folosind ipotezele $1^* - 5^*$ modelul foloseşte funcţia obiectiv

$$\max(R = \sum_{i=1}^{K} F_k \prod_{i \in S_k} R_i) \tag{4.34}$$

unde

$$R_i = 1 - \prod_{j=1}^{m_i}(1 - R_{ij}$$

cu restricţiile

$$(i) \quad \sum_{i=1}^{m_i} X_{ij} \geq 1, \quad i = 1, 2, ..., n$$

$$(ii) \quad \sum_{i=1}^{n} \sum_{j=1}^{m_i} X_{ij} C_{ij} \leq B, \quad X_{ij} \in \{0, 1\}, \quad \forall i = 1, 2, ..., n, j = 1, 2, ..., m_i.$$

Acesta este un model de programare neliniară, dinamică, discretă, cu restricţii inegalităţi. El este de tipul modelului • • .

4.3 Alte modele relative la costuri

In această subsectiune vom prezenta pe scurt câteva modele reprezentative bazate pe costuri [19], care s-au impus prin faptul că s-au dovedit utile în aplicaţiile practice.

4.3.1 Modele de tip COCOMO.

Denumirea acestor modele provine de la *COnstructive COst MOdel*. COCOMO este un sistem computerizat care constă de fapt din trei modele: COCOMO principal, COCOMO intermediar şi COCOMO detailat. Fiecare din acestea include un număr de algoritmi care leagă *volumul* produsului soft exprimat in mii de linii (de program sursă), notat DSI, de efortul de realizare exprimat în oameni-luni MM_{mon}. Expresiile intermediare ale COCOMO sunt rafinate de un număr de corecţii constând din factori multiplicativi care iau valori specifice tipului de proiecte depinzând de factori ce influenţează costul.

Toţi algoritmii privind efortul de realizare au următoatea formă (*de bază*)

$$MM_{mon} = \alpha(DSI)^{\beta} \tag{4.35}$$

unde estimarea efortului nu include cheltuielile colaterale legate de studiile de fezabilitate, de analiză a condiţiilor logistice sau de întreţinere, adică se consideră numai efortul concentrat pe proiect.

Coeficienţii α, β depind de sistemul COCOMO utilizat (adică principal, intermediar sau detailat) şi de tipul de proiect. In literatură sunt utilizate valorile din tabelul de mai jos, unde tipurile de proiecte sunt clasificate in

a) *organic,* adică un proiect software mic realizat pentru scopuri restrânse (utilizat în firme mici cu cerinţe reduse);

c) *elaborat,* adică proiect soft ccare satisface cerinţe complexe pentru clase mari de beneficiari, utilizând memorie mare şi consum mare de timp;

b) *semi-detaşat,* adică intermediar intre cele două tipuri extreme.

Model→	Principal		Intermediar & Detaliat	
Tip proiect ↓	α	β	α	β
Organic	2.4	1.05	3.2	1.05
Semi-detaşat	3.0	&1.12	3.0&	1.12
Elaborat	3.6	1.20	2.8	1.20

COCOMO este prevăzut cu tabele care precizează repartizarea efortului pe fazele de dezvoltare a softului şi pe activităţile specifice fiecărei faze. Folosind aceste tabele, efortul estimat conform modelului este divizat pe activităţi în raport cu care se planifică realizarea proiectului.

Modelele **intermediare şi detaliate** permit estimarea efortului folosind factori multiplicativi de corecţie faţă de un proiect "mediu". Sunt 15 categorii de corecţii de costuri utilizate de cele două modele (intermediar şi detaliat) care se referă la: atributele produsului (fiabilitate, marimea bazei de date, complexitatea softului); atributele calculatorului pe care va fi utilizat produsul soft (restricţii privind timpul de execuţie şi memoria utilizată, etc); atributele personalului (capabilitatea de analiză, experienţa de proiectare, experienţa de programare, cunoaşterea limbajelor de programare, etc.); atributele proiectului (utilizarea practicilor şi facilităţilor moderne de programare, utilizarea instrumentelor soft, utilizarea tehnicilor moderne de conducere şi monitorizare a proiectelor). Costurile sunt clasificate in 5 nivele

de importanţă (foarte mic, mic, normal, mare, foarte mare). In [19] se găseşte un tabel cu valorile acestor corecţii multiplicative de cost care se utilizează astfel: de ex. pentru a corecta valoarea lui MM_{mon} se ia valoarea *nominală* (reală) a lui M_{mon} şi se înmulţeşte cu coeficientul corespunzător din tabel. Factorii de corecţie sunt crescători sau descrescători în funcţie de nivelele de importanţă. De ex.utilizarea de instrumente soft deja existente la un nivel mic presupune un factor de coreţie mare, pe când cerinţa de realizare a unui soft cu fiabilitate mică presupune un factor de corecţie mic. Valorile concrete ale factorilo de corecţie, ca de altfel toate valorile coeficienţilor din modelele COCOMO au fost estimate din date istorice.

Modelele de tip COCOMO permit de asemenea estimarea eforturilor legate de *re-utilizarea, (adaptarea)* unor părţi din produsele soft realizate anterior prin calcularea unui *număr echivalent de instrucţiuni sursă livrate* ($EDSI$) şi utilizarea acestora în locul lui DSI în (4.35). Astfel $EDSI$ se calculează in funcţie de DSI prin formula

$$EDSI = DSI \times AAF/100, \tag{4.36}$$

cu

$$AAF = 0.40 \times DM + 0.30 \times CM + 0.30 \times IM, (medie \quad ponderata), \tag{4.36'}$$

unde DM este procentul în care se modifică proiectul deja existent, CM este procentul în care se modifică codul sursă al softului, iar IM este procentul efortului necesar ca să se integreze softul adaptat in sistemul nou impreună şi cu testarea softului rezultat.

O componentă importantă a modelelor COCOMO este şi aceea a estimării **efortului de întreţinere** al produselor soft, dar restricţionat numai la reproiectarea unor părţi ale produsului soft, proiectarea unor pachete mici de interfaţă care cer reproiectarea unei mici părţi a produsului (sub 20%!), precum şi modificarea codului sursă, a documentaţiei sau bazei de date. Efortul anual de intreţinere MM_{AM} este dat de formula

$$MM_{AM} = ACT \times MM_{mon} \tag{4.37}$$

unde ACT este *traficul anual al modificărilor,* adică o fracţiune din instrucţoinile ce ar urma să se modifice de-alungul unui an prin adăugarea de modificări.

Modelul principal COCOMO se aplică la proiectele convenţionale care folosesc aceleaşi criterii de determinare a efortului DSI în timp ce modelele

intermediare sau detaliate se folosesc in alte cazuri care presupun corecţii legate de *combinarea* calculelor de estimare a eforturilor.

Detalii privind aplicaţiile practice şi acurateţea modelelor de tip COCOMO se găsesc in [19] unde se rezumă o serie de sugestii şi recomandări culese din literatura de specialitate.

4.3.2 Modelul Walston-Felix

Acest model, prezentat in [19], construit în 1977 cu date culese în anii '60 de la firma IBM estimează efortul E în funcţie de volumul S al produsului soft pe baza formulei

$$E = \alpha S^{\beta}. \tag{4.38}$$

Folosind transformarile $Y = \log E, A = \log \alpha, X = \log S$ se obţine ecuaţia de regresie liniară

$$Y = A + \beta X \tag{4.38'}$$

iar cu metoda celor mai mici pătrate se estimează β şi A precum şi $\alpha = e^{A}$. Folosind datele istorice menţionate, Walston şi Felix au găsit formula estimativă

$$\hat{E} = 5.2S^{0.91}. \tag{4.38''}$$

Ultima formulă reprezintă o *medie* a efortului în raport cu volumul S al programului. Aplicat în cazuri particulare modelul prezintă abateri ce depind de un număr mare de factori ce influenţează *indicele* I al *productivităţii* fiecărui proiect calculat cu formula

$$I = \sum_{i=1}^{nf} W_i X_i \tag{4.39}$$

unde nf este numărul facorilor. X_i are semnificaţia

$$X_i = \begin{cases} 1 & daca\ \ factorul\ \ i\ \ creste\ \ productivitatea \\ 0 & daca\ \ factorul\ \ i\ \ este\ \ neinfluentabil \\ -1 & daca\ \ factorul\ \ i\ \ micsoreaza\ \ productivitatea \end{cases}$$

iar

$$W_i = \frac{1}{2}\log(\Delta L_i) \tag{4.39'}$$

şi ΔL_i reprezintă *rata de variaţie a productivităţii lui* i. In acest caz estimarea efortului in raport cu productivitatea este

$$E = \frac{S}{L}$$

unde S este volumul programului exprimat în număr de linii de cod-sursă.

Pentru proiecte soft de mare complexitate, în care intervin tot mai mulţi factori care influenţează productivitatea, formula (4.39') se îmbunătăţeşte dacă se schimbă scalele de valori şi anume

$$E = S/P \tag{4.40}$$

unde S este numărul *miilor* de linii-cod iar P este ***productivitatea medie a proiectului*** exprimată în ***kilo-linii per om-lună.***

4.4 Metamodelul Bailey-Basili

In [19] se prezintă următorul model, care cu notaţiile de mai sus se bazează pe relaţia

$$E = \alpha + \beta S^{\gamma} \tag{4.41}$$

unde α este interpretat ca timpul iniţial necesar pentru a înţelege proiectul inainte de a începe programarea.

Folosind date istorice pentru un număr de proiecte dat N, Bailey şi Basili au estimat parametrii minimizând expresia sumei pătratelor erorilor standard ale estimaţiilor

$$SSE = \sum_{i=1}^{N} \left[1 - \frac{\alpha + \beta S_i^{\gamma}}{E_i}\right]^2 \tag{4.42}$$

care in final a condus la relaţia

$$\hat{E} = 3.5 + 0.73 S^{1.16} \tag{4.42'}$$

dar cu o eroare $SSE = 1.25$ destul de mare. Pentru a reduce această eroare se foloseşte *factorul multiplicativ de ajustare* E_{adj} ce leagă eforturile estimate $\hat{E}_i$ de valorile observate E_i prin formula

$$ER_{adj} = \begin{cases} R - 1 & daca \quad R > 0 \\ 1 - \frac{1}{R} & daca \quad R < 0, \end{cases} \tag{4.43}$$

unde $R = E/\hat{E}$. Efortul corectat satiface deci ecuaţia

$$E = \begin{cases} (1 + ER_{adj})\hat{E} & daca \quad ER_{adj} > 0 \\ \hat{E}(1 + |ER_{adj}|) & daca \quad ER_{adj} < 0. \end{cases} \tag{4.44}$$

Pentru a îmbunătăţi estimaţia efortului s-a propus o extinderee care ia în consideraţie alţi factori ce influenţează costul proiectului cum ar fi metodologia de proiectare-programare, complexitatea proiectului şi experienţa personalului proiectant. Modelul care exprimă efortul in acest caz este dat de ecuaţia de regresie multiplă

$$ER_{adj} = \alpha + \beta METH + \gamma CPLX + \delta EXP \tag{4.45}$$

unde $METH$ este măsurată de o scală de nouă valori între 0 şi 5 care corespunde celor trei tipuri de scheme (logice,diagrame de structura şi diagrame de sistem), proiectării top-down, documentării formale, echipei programatorului şef, planurilor formale de testare,etc. Mărimea $CPLX$ este complexitatea măsurată de o scală de şapte valori măsurate tot între 0 şi 5 (care include caracteristici ale complexităţii interfeţelor cu utilizatorul, complexitatea aplicaţiilor, complexitatea fluxului de programe, complexitatea comunicării interne, complexitatea bazei de date, complexitatea comunicării externe şi complexitatea modificărilor ce pot fi făcute de utilizatori). Mărimea EXP este valoarea experienţei personalului, exprimată de o scală de cinci valori tot între 0 şi 5 (care include calificarea programatorilor, experienţa de utilizare a computerului de către personal, experienţa personalului de a dezvolta aplicaţii şi de a lucra în echipă).

In aplicaţii se recomandă ca efortul de realizare a softului E să fie determinat atât cu formula (4.42') cât şi cu formula (4.45).

4.5 Modele de tip Putnam

In această categorie intră modelele care se ocupă de repartiţia (în timp) a efortului uman consumat in producţia de software. Vom prerzenta mai întâi modelul original al lui Putnam [19], iar apoi câteva versiuni de interes practic.

O curbă care indică numărul de persoane implicate în procesul de proiectare-implementare a unui produs soft în raport cu timpul are forma unui clopot asimetric ce s-a dovedit a fi aproape identic cu graficul densităţii de repartiţie de tip Rayleygh (Fig.4.1). Această repartiţie satisface ecuaţia diferenţială

$$\frac{dy}{dt} = 2Kate^{-at^2} \tag{4.46}$$

unde dy/dt este rata de variaţie în timp a forţei de muncă, t este timpul scurs de la începerea proiectului, a este o constantă ce determină forma curbei, iar

K este o constantă de normare (a densităţii). Integrând se obţine

$$y(t) = K(1 - exp(-at^2)) \tag{4.46'}$$

unde $y(t)$ este numărul cumulat al personalului utilizat până la momentul t.

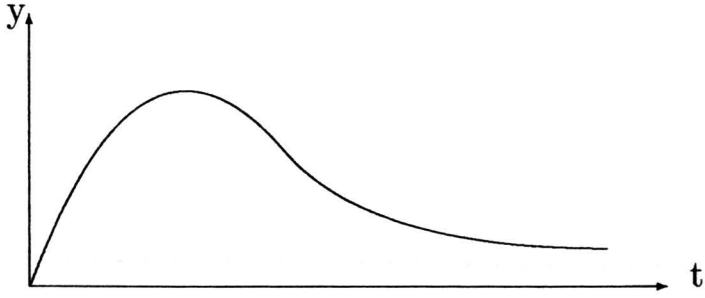


Fig.4.1 Curba lui Rayleigh.

Din (4.46') se observă că dacă $t \to \infty$ atunci $y \to K$, deci K reprezintă efortul total cumulat (în termeni de personal) de-alungul ciclului de viaţă al prdusului soft inclusiv efortul de întreţinere. In efortul uman consumat nu intră studiile de fezabilitate, de analiză colaterală in vederea realizării proiectului. Formula (4.46') sugerează că timpul necesar elaborării primei versiuni a produsului soft este momentul t_d care corespunde maximului lui $y(t)$, adică

$$t_d = \left(\frac{1}{2}a\right)^{1/2} \tag{4.47}$$

obţinut ca soluţie a ecuaţiei $y'(t) = 0$. De aici rezultă că efortul uman E pentru livrarea primei versiunu a softului este

$$y(t_d) = k[1 - e^{-1/2}] \approx 0.349K = E. \tag{4.47'}$$

Se pot considera câteva *extensii* ale modelului lui Putnam.Astfel, s-a observat că proiectele realizate cu o productivitate mare folosesc la inceput o rată aforţei de muncă mai redusă în timp ce proiectele cu productivitate mică folosesc ilustrează o rată de creştere mare a personalului la început. Această rată iniţială a forţei de muncă este interpretată ca *dificultatea* D a proiectului şi se obţine din diferenţierea lui (4.46) în punctul t_d, adică

$$D = \frac{K}{t_d^2}. \tag{4.48}$$

Putnam a presupus că există o legătură între dificultatea D şi productivitatea L de forma

$$L = \alpha D^{\beta} \tag{4.47'}$$

iar pe de-altă parte

$$L = \frac{S_s}{E} \tag{4.48'}$$

unde S_s este numărul de mii de linii cod (program) produse. Folosindu-se metoda regresiei neloiniare s-a găsit că $\beta = -2/3$ de unde din (4.48') se deduce

$$S_s = \alpha D^{-2/3}(0.349K). \tag{4.49}$$

Din relaţiile precedente se deduce că

$$S_s = C\left[\frac{K}{t_d^2}\right]^{-2/3} K = CK^{1/3}(t_d)^{4/3} \tag{4.50}$$

unde C este interpretat ca *factor de tehnologie.*

Formula (4.50) poate fi considerată ca relaţie fundamentală între efort şi timp. Pentru diferite proiecte s-au stabilit 20 valori ale lui C variind de la 610 la 57314, valori ce depind de factori ca: restricţii de hardware, experienţa personalului şi mediul de programare. In plus Putnam a propus următoarea relaţie

$$D_0 = \frac{K}{t_d^3} \tag{4.51}$$

unde D_0 este o constantă interpretată ca *acceleraţia forţei de muncă* ce poate lua valori discrete depinzând de tipul proiectului şi anume: $D_0 = 7.3$ pentru un produs soft nou cu multe interfeţe ce interaţionează cu un alt sistem; $D_0 = 15$ pentru sisteme noi de sine stătătoare; $D_0 = 27$ pentru sisteme reproiectate. In total sunt propuse şase valori pentru D_0. Formula lui D_0 s-a obţinut prin derivarea lui (4.48), iar relaţia a fost testată empiric pe un număr de proiecte particulare.

Relaţia (4.51) este esenţială pentru estimări deoarece combinându-o cu (4.50) se obţin formulele

$$T_d = \left[\frac{S_s^3}{D_0 C^3}\right]^{1/7} \tag{4.52}$$

$$K = \left(\frac{S_s}{C}\right)^{9/7} (D_0)^{4/7} \tag{4.53}$$

care se referă numai la un singur parametru legat de costuri, K sau t_d.

• **Modelarea ciclului de viaţă al componentelor.** Formula lui Raileygh se utilizează şi pentru fiecare fază a ciclului de viaţă al proiectului (de ex. definirea specificaţiilor, proiectare şi codificare, testare şi validare, întreţinere etc.) astfel că formula referitoare la întreg proiectul devine rezultatul "combinării" unor formule referitoare la faze. De exemplu, pentru proiectare şi codificare (faza cea mai importantă a execuţiei proiectului) rata efortului uman se exprimă tot printr-o relaţie de tip (4.46)

$$\frac{dy_1}{dt} = 2K_1 bt \exp -bt^2 \tag{4.54}$$

unde K_1 este efortul total cumulat pe fază iar

$$b = \frac{1}{2(t_{1d})^2}$$

analog lui (4.47). Deoarece dezvoltarea fazei începe odată cu dezvoltarea proiectului rezultă că

$$\frac{K}{t_d^2} = \frac{K_1}{t_{1d}^2}. \tag{4.55}$$

Putnam face ipoteza că timpul de dezvoltare al proiectului t_d corespunde la 95% din timpul total de proiectare şi codificare de unde rezultă

$$1 - \exp\{-bt_d^2\} = 0.95$$

iar dacă ţinem seama de valoarea lui b avem

$$0.05 = \exp\{-\frac{t_d^2}{2t_{1d}^2}\}. \quad 2\log 0.05 = \left[\frac{t_d}{t_{1d}}\right]^2, \quad \frac{t_d}{t_{1d}} \approx 6^{1/2}. \tag{4.56}$$

Folosind (4.56) relaţia (4.55) devine

$$K_1 = \frac{K}{6}. \tag{4.55'}$$

Se poate constata că o consecinţă a relaţiei (4.50) este că mărimea K variază invers in rapport cu puterea a patra a lui t_d, adică

$$K = \left(\frac{S_s}{C}\right)\frac{1}{t_d^4}. \tag{4.57}$$

Pentru estimarea lui K din (4.57) ar trebui estimat mai întâi C, S_s şi t_d precum şi D_0 care a fost discutat mai sus.

• **Versiunea lui Parr.** Ipoteza că mărimea forţei de muncă (efortului) pentru realizarea unui produs soft ar avea forma unei curbe Raileygh (ce porneşte de la zero, creşte în timp şi apoi scade la zero), a fost criticată din considerente practice. S-a propus [19] să se folosească, mai ales în vecinătatea momentului $t = 0$ relaţia

$$\frac{dy}{dt} = m(t) = \frac{1}{4} sech^2[\frac{at + c}{2}] \tag{4.58}$$

unde a şi c sunt parametri ce determină forma curbei. Modelul Raileygh şi modelul Parr (4.58) se aplică impreuna; pe un interval $[0, t_0]$ se aplică relaţia lui Parr (care la momentul iniţial nu mai corespunde valorii zero a ratei de variaţie a efortului uman!), iar apoi, pe (t_0, ∞) se aplică relaţia lui Raileygh (4.46).

• **Versiunea lui Jensen.** Acest model [19] presupune că relaţia de bază este

$$S_s = C_{tc} t_d K^2 \tag{4.59}$$

unde în afară de notaţiile deja introduse intervine *constanta de eficienţă tehnologică* C_{tc} dat de relaţia

$$C_{tc} = C_{tb} \prod_{i=1}^{ls} f_i \tag{460}$$

unde fi este măsura costului activităţii i a modelului COCOMO.

4.6 Versiuni ale modelului COCOMO sau asemănătoare

• **Modelul COCOMO pentru planificare.** Conform COCOMO, timpul de dezvoltare *planificat* al proiectului $TDEV$ este dat de relaţia

$$TDEV = \alpha(MM)^{\beta} \tag{4.61}$$

unde MM, măsurat în oameni-luni, este efortul uman utilizat pentru a realiza produsul soft; constantele sunt $\alpha = 2.5$, iar β depinde de tipul de model COCOMO folosit (organic, semidetaşat, elaborat) şi ia respectiv valorile 0.38; 0.35; 0.32.

•**Modelul COPMO.** Acest model descrie relaţiile dintre necesarul de personal, efortul global şi mărimea proiectului. El (COPMO) înseamnă CO-operative Programming MOdel. Numărul personalului P_a se exprimă astfel

$$P_a = \frac{E}{T} \tag{4.62}$$

unde E este efortul exprimat în oameni-luni şi T este durata exprimată în luni. Pentru calculul efortului se propune formula

$$E = E_P(S) + E_c(P_a) \tag{4.63}$$

unde E_P este efortul necesar pentru una sau mai multe persoane care lucrează independent la realizarea de module care nu necesită interacţiuni cu alte module, S este volumul total al softului exprimat în mii de linii de cod (program), E_c este efortul cerut de coordonarea activităţilor programatorilor din echipă, iar P_a este mărimea medie a echipei dată de (4.62). In plus se presupune că

$$E_P(S) = a + bS, \quad E_c(P_a) = c(P_a)^d, \tag{4.64}$$

adică în final (4.63) devine

$$E = a + bS + c(P_a)^d. \tag{4.55}$$

Estimarea parametrilor a, b, c, se face cu ajutorul *metodei celor mai mici pătrate în două etape* şi anume:

- se determină parametri $\hat{a}.\hat{b}$ cu metoda regresiei liniare folosind date istorice pentru care $P_a \approx 1$;
- cu $\hat{a}, \hat{b}$ determinaţi, folosind date istorice pentru care $P_a \neq 1$ se determină $c.d$ din modelul de regresie liniarizabil

$$E - \hat{E}_P(S) = c(P_a)^d. \tag{4.66}$$

• **Modelul lui Jeffery.** O alternativă a modelului COPMO [19] se datorează lui Jeffery şi are forma

$$L = aS^b M^{-c} \tag{4.67}$$

unde S este ca mai înainte volumul, L este productivitatea ($L = E/T$) iar nivelul maxim al personalului este M. Parametri a, b, c ai acestui model se

estimează cu uşurinţă prin metoda celor mai mici pătrate deoarece modelul este liniarizabil.

Modelul COPMO şi versiunea lui Jeffery evidenţiază în aplicaţii că efortul creşte când nivelul mediu al personalului creşte.

După câte se observă, modelele acestei ultime secţiuni sunt importante pentru planificarea şi urmărirea practică a execuţiei proiectelor soft.

Totuşi, în aplicaţiile practice nu trebuie să se ia decizii numai pe baza unui singur model, deoarece nici ipotezele modelului respectiv nu pot fi în totalitate satisfăcute şi nici datele de intrare ale modelului nu pot fi culese fără a fi afectate de erori mai ales intâmplătoare. De aceea pentru modelele utilizate, care estimează aceleaşi caracteristici de complexitate, de calitate sau de fiabilitate, precum şi care estimează parametri statistici, este necesar să se calculeze (estimeze), în măsura posibilităţilor, erori medii pătratice sau erori relative, care pot da informaţii despre acurateţea modelelor. Pe baza acestor erori medii pătratice se poate alege modelul cel mai bun (cu cea mai mică eorare) in cazul concret dat.

In final să observăm că modelele bazate pe costuri folosesc şi *metricile de complexitate* a programelor ce vor fi prezentate in capitolul 6.

5 Modele Bayesiene pentru fiabilitatea programelor

5.1 Introducere in analiza Bayessiană

De obicei modelele Bayessiene se construiesc şi utilizează atunci când se cunoaşte de la început ceva despre parametri ce trebuie estimaţi, sau cum se mai spune, se cunoaşte o *informaţie apriorică* despre aceşti parametri [6,8,12,24,35]. Să presupunem că parametrul real este θ, cu $\theta \in \Omega \subset \mathbf{R}$. Se presupune că informaţia asupra lui θ este cunoscută in sensul că θ este o variabilă aleatoare a cărei densitate de rerpartiţie (dr) este o funcţie $g(\theta)$ cunoscută, numită *densitate de repartiţie apriori.* Dacă $t_i, 1 \leq i \leq n, t_i = T_i - T_{i-1}$ sunt datele de observaţie asupra duratelor de funcţionare, (amintim că T_i sunt momentele căderilor pe axa timpului operaţional) atunci notând $\bar{t} = (t_1, t_2, ..., t_n)'$ vectorul de selecţie şi cu $f(t|\theta)$ densitatea de repartiţie a duratei in funţionare, rezultă că funcţia de verosimilitate (numită *verosimiltate apriorică*) este

$$f(\bar{t}|\theta) = \prod_{i=1}^{n} f(t_i|\theta). \tag{5.1}$$

Atunci *funcţia de verosimilitate aposteriorică* este

$$f(\bar{t}|a, b, ...) = \int_{-\infty}^{\infty} f(t|\theta) g(\theta, a, b, ...) d\theta, \tag{5.1'}$$

unde $a, b, ...$ sunt parametri de care depinde densitatea de repartiţie apriorică g a lui θ. Se observă că funcţia de verosimilitate aposteriorică este *mixtura* sau *amestecarea* funcţiei de verosimilitate apriorică cu densitatea de repartiţie apriorică a lui θ. Conform formulei lui Bayes *funcţia de densitate aposteriorică* a lui θ este

$$h(\theta|\bar{t}) = \frac{f(\bar{t}|\theta) g(\theta)}{\int_{\Omega} f(\bar{t}|\theta) g(\theta) d\theta}. \tag{5.2}$$

In analiza bayesiană, când se caută o estimaţie (numită *estimaţie bayessiană*) $\hat{\theta}$ a lui θ, se consideră o *funcţie de pierdere* $l(\theta, \hat{\theta})$ care este o măsură a *importanţei erorii de estimare.* A determina estimaţia bayesiană $\hat{\theta}$ a lui

θ inseamnă a determina această estimaţie *minimizând pierderea medie în raport cu densitatea de repartiţie aposteriory*, adică minimizând funcţia

$$E[l(\theta,\hat{\theta})] = \int_{\Omega} l(\theta'\hat{\theta})h(\theta|\bar{t})d\theta,\ \theta \in \Omega. \tag{5.3}$$

Funcţia de pierdere $l(\theta,\hat{\theta})$ poate fi aleasă în mai multe moduri; două dintre acestea vor fi considerate în cele ce urmează.

• **Funcţia de pierdere pătratică.** Aceasta este de forma

$$l(\theta,\hat{\theta}) = c(\hat{\theta}-\theta)^2,\ \theta \in \Omega \subset R \tag{5.4}$$

unde c este o constantă de proporţionalitate. Condiţia de minim este

$$\frac{dE[l(\theta,\hat{\theta})]}{d\theta} = 2c\int_{\Omega}(\hat{\theta}-\theta)h(\theta|\bar{t})d\theta = 0$$

care în final devine

$$\hat{\theta} = \int_{\Omega}\theta(\theta|\bar{t})d\theta \tag{5.4'}$$

adică estimaţia $\hat{\theta}$ este media lui θ in raport cu repartiţia sa aposteriorică.

• **Funcţia de pierdere-valoare absolută.** Aceasta este de forma

$$l(\theta,\hat{\theta}) = |\theta-\hat{\theta}|,\ \theta \in \Omega = R. \tag{5.5}$$

Condiţia de minim conduce la $\hat{\theta} = mediana$, adică $\hat{\theta}$ satisface condiţia

$$\int_{-\infty}^{\hat{\theta}} h(\theta|\bar{t})d\theta = \int_{\hat{\theta}}^{\infty} h(\theta|\bar{t})d\theta. \tag{5.5'}$$

In concluzie, pentru a efectua analiza bayesiană asupra unui parametru θ ce trebuie estimat, este necesar mai întâi să ştim *densitatea de repartiţie aprioricâ* $g(\theta)$ a lui θ, apoi să determinăm *densitatea de repartiţie aposteriorică* $h(\theta|\bar{t})$ a lui θ iar în final să determinăm media sau mediana lui θ în raport cu repartiţia aposteriorică. Aceste ultime elemente $\hat{\theta}$ (adică media sau mediana) constituie *estimaţiile bayasiene* ale lui θ.

• **Media aposteriorică.** Dacă în loc de densitatea aposteriorică $h(\theta|\bar{t})$ a lui θ considerăm funcţia de verosimilitate aposteriotică de forma $g(\bar{t}|a,b,...)$

(depinzând de parametri necunoscuţi), atunci ne putem pune problema estimării parametrilor $a, b, ...$ Aceşti parametri se estimează de obicei cu metoda verosimilităţii maxine (să-i notăm $\hat{a}, \hat{b}, ..$) iar estimaţia bayessiană este în acest caz valoarea medie *aposteriorică* $\overline{\theta}$ a lui θ, adică

$$\overline{\theta} = \int_{\Omega} \theta g(\theta, \hat{a}, \hat{b}, ...) d\theta. \tag{5.5'}$$

In cele ce urmează vom studia versiunile bayesiene ale unor modele clasice de fiabilitatea programelor, tratate în capitolele 2 şi 3.

5.2 Modelul lui Littlewood-Veral

Presupunem [24,35] că intervalul de timp t_i dintre a $(i-1)-$a şi a $i-$a cădere are repartiţie exponenţială de parametru $\lambda_i, 1 \leq i \leq N_0$, adică are d.r.

$$f(t_i|\lambda_i) = \lambda_i exp(-\lambda_i t_i). \tag{5.6}$$

Presupunem de asemenea că parametrul λ_i este aleator şi are repartiţia apriori de tip $Gamma(0, \Psi(i), \alpha)$, adică are *dr*

$$g(\lambda_i|\alpha, \Psi(i)) = \frac{[\Psi(i)]^{\alpha}}{\Gamma(\alpha)} \lambda_i^{\alpha-1} e^{-\Psi(i)t_i} \tag{5.7}$$

unde α este parametrul *de formă* iar $\Psi(i)$ este parametrul de scală care depinde de eroarea a $i-$a detectată. Ultimul parametru descrie deobicei calitatea testării şi este crescător in i. Din acest fapt se deduce că λ_i este *stochastic descrescător* in i, adică

$$P(\lambda_i \leq \lambda) \geq P(\lambda_{i-1} \leq \lambda).$$

Densitatea de repartiţie aposteriori a lui t_i când se dă α şi $\Psi(i)$ este

$$f(t_i|\alpha, \Psi(i)) = \int_0^{\infty} f(t_i|\lambda_i) g(|\lambda_i, \Psi(i)) d\lambda_i =$$

$$= \int_0^{\infty} \lambda_i exp(-\lambda_i t_i) \frac{[\Psi(i)]^{\alpha} \exp(-\Psi(i)\lambda_i)}{\Gamma(\alpha)} d\lambda_i = \alpha \frac{[\Psi(i)]^{\alpha}}{[t_i + \Psi(i)]^{\alpha+1}}.$$

Deci funcţia de repartiţie aposteriori a lui t_i când se dă α şi $\Psi(i)$ este

$$F(t_i|\alpha, \Psi(i)) = 1 - \left[\frac{\Psi(i)}{t_i + \Psi(i)}\right]^{\alpha} \tag{5.8}$$

de unde rezultă că *rata căderilor aposteriori* este

$$\lambda(t_i|\alpha, \Psi(i)) = \frac{\alpha}{t_i + \Psi(i)}. \tag{5.8'}$$

Se observă că în ipotezele acestui model, timpul dintre căderi are o repartiţie aposteriori de tip *Pareto* care este o *repartiţie de tip DFI.*

Funcţia de verosimilitate aposteriorică a selecţiei $t_1, t_2, ..., t_n$ este

$$f(t_1, t_2, ..., t_n|\alpha, \Psi(i)) = \frac{\alpha^n \prod_{i=1}^{n} [\Psi(i)]^{\alpha}}{\prod_{i=1}^{n} [t_i + \Psi(i)]^{\alpha+1}}. \tag{5.9}$$

Pentru a duce până la capăt aplicarea metodei verosimilităţii (aposteriorice) maxime vom considera câteva forme particulare ale lui $\Psi(i)$.

• **Cazuri particulare.** Presupunem [24,25,35] mai intâi că

$$\Psi(i) = \beta_0 + \beta_1 i$$

unde β_0 şi β_1 sunt parametri ce trebuie estimaţi. Atunci funcţia de verosimilitate aposteriorică devine

$$L(t_1, t_2, ..., t_n|\alpha, \beta_0, \beta_1) = \frac{\alpha^n \prod_{i=1}^{n} [\beta_0 + \beta_1 i]^{\alpha}}{\prod_{i=1}^{n} [t_i + \beta_0 + \beta_1 i]^{\alpha+1}}. \tag{8.10}$$

Condiţiile de maxim pentru funcţia de verosimilitate aposteriorică conduc la următorul sistem de ecuaţii in α, β_0, β_1

$$\frac{\partial \log L}{\partial \alpha} = n + \sum_{i=1}^{n} \alpha[\log(\beta_0 + \beta_1 i) - \log(t_i + \beta_0 + \beta_1 i)] = 0$$

$$\frac{\partial \log L}{\partial \beta_0} = \sum_{i=1}^{n} \left[\frac{\alpha}{\beta_0 + \beta_1 i} - \frac{\alpha + 1}{t_i + \beta_0 + \beta_1}\right] = 0 \tag{5.10}$$

$$\frac{\partial \log L}{\partial \beta_1} = \sum_{i=1}^{n} \left[\frac{\alpha i}{\beta_0 + \beta_1 i} - \frac{(\alpha+1)i}{t_i + \beta_0 + \beta_1 i} \right] = 0.$$

Eliminând pe α din cele trei ecuaţii obţinem următoarele două ecuaţii în β_0 şi β_1

$$\frac{n}{\sum_{i=1}^{n}[\log(t_i + \beta_0 + \beta_1 i) - \log(\beta_0 + \beta_1)]} = \frac{\sum_{i=1}^{n} \frac{1}{t_i+\beta_0+\beta_1 i}}{\sum_{i=1}^{n} \left[\frac{1}{\beta_0+\beta_1 i} - \frac{1}{t_i+\beta_0+\beta_1 i}\right]}$$

$$\frac{\sum_{i=1}^{n} \frac{1}{t_i+\beta_0+\beta_1 i}}{\sum_{i=1}^{n} \left[\frac{1}{\beta_0+\beta_1 i} - \frac{1}{t_i+\beta_0+\beta_1 i}\right]} = \frac{\sum_{i=1}^{n} \frac{i}{t_i+\beta_0+\beta_1 i}}{\sum_{i=1}^{n} \left[\frac{i}{\beta_0+\beta_1 i} - \frac{i}{t_i+\beta_0+\beta_1 i}\right]}.$$

Dacă punem aceste ecuaţii sub forma

$$\beta_0 = f_0(\beta_0, \beta_1, \bar{t}), \quad \beta_1 = f_1(\beta_0 + \beta_1, \bar{t}) \tag{5.11}$$

şi considerăm vectorul coloană $\mathbf{f} = (f_0, f_1)'$ şi $\beta = (\beta_0, \beta_1)'$ atunci sistemul de ecuaţii capătă următoarea formă vectorială

$$\beta = \mathbf{f}(\beta, \bar{t}). \tag{5.11'}$$

Pornind cu o aproximare iniţială $\beta^{(0)}$ o soluţie numerică β^* se obţine prin următorul procedeu iterativ

$i := 0$
repeat
$\beta^{i+1} = \mathbf{f}(\beta^i, \bar{t})$
until $||\beta^{i+1} - \beta^i|| \leq \epsilon$

unde ϵ este un număr pozitiv, suficient de mic (eroarea de aproximaţie).Soluţia β^* a sistemului este media celor două valori β^i, β^{i+1} sau oricare dintre ele.

Altă formă a lui $\Psi(i)$ poate fi

$$\Psi(i) = \exp(\beta_0 + \beta_1 i).$$

In acest caz *funcţia de logverosimilitate aposteriorică* este

$$l = \log L = n \log \alpha + \alpha \sum_{i=1}^{n} (\beta_0 + \beta_1 i) - \sum_{i=1}^{n} (\alpha+1) \log(t_i + \exp(\beta_0 + \beta_1 i)). \tag{5.12}$$

Condiţiile de optim (adică de maximum al verosimilităţii în vederea estimării parametrilor α, β_0, β_1) conduc la următorul sistem de ecuaţii

$$\frac{n}{\alpha} + \sum_{i=1}^{n}(\beta_0 + \beta_1 i) - \sum_{i=1}^{n}\log(t_i + \exp(\beta_0 + \beta_1 i)) = 0$$

$$\alpha n - (\alpha + 1)\sum_{i=1}^{n}\frac{\exp(\beta_0 + \beta_1 i)}{t_i + \exp(\beta_0 + \beta_1 i)} = 0 \qquad (2.13)$$

$$\alpha\sum_{i=1}^{n} i - (\alpha + 1)\sum_{i=1}^{n}\frac{i\exp(\beta_0 + \beta_1 i)}{t_i + \exp(\beta_0 + \beta_1 i)} = 0.$$

Eliminând pe α între a treia ecuaţie şi prima ecuaţie obţinem următorul sistem de ecuaţii neliniare in parametri β_0, β_1

$$n^2 = \left(\sum_{i=1}^{n}\log(t_i + \exp(\beta_0 + \beta_1 i) - \sum_{i=1}^{n}(\beta_0 + \beta_1 i) + n\right) \cdot \sum_{i=1}^{n}\frac{\exp(\beta_0 + \beta_1 i)}{t_i + \exp(\beta_0 + \beta_1 i)};$$

$$\left[\frac{n(n+1)}{2} - \sum_{i=1}^{n}\frac{\exp(\beta_0 + \beta_1 i)}{t_i + \exp(\beta_0 + \beta_1 i)}\right] \cdot \sum_{i=1}^{n}\frac{\exp(\beta_0 + \beta_1 i)}{t_i + \exp(\beta_0 + \beta_1 i)} =$$

$$= \left[n - \sum_{i=1} 6\frac{\exp(\beta_0 + \beta_1 i)}{t_i + \exp(\beta_0 + \beta_1 i)}\right] \cdot \sum_{i=1}^{n}\frac{iexp(\beta_0 + \beta_1 i)}{t_1 + \exp(\beta_0 + \beta_1 i)}.$$

Acest sistem poate fi rezolvat numeric într-un mod asemănător sistemului (5.11'). Odată ce estimaţiile lui β_0, β_1 sunt calculate, estimaţia lui α se poate determina de exemplu din cea de-a doua ecuaţie (2.13).

• **Versiunea lui Musa.** Musa [25] presupune că $\Psi(i)$ este de forma

$$\Psi(i) = \frac{N_0 T_0 \alpha}{N_0 - i} \qquad (5.14)$$

unde N_0 este numărul mediu de erori din soft pe perioada sa de viaţă, T_0 este valoarea medie a timpului iniţial de cădere şi α este parametrul de *formă* al repartiţiei apriorice gamma a lui λ_i.

Funcţia de verosimilitate aposteriorică a lui $\bar{t}$ când se dă N_0, T_0, α este

$$L(\bar{t}|\alpha, N_0, T_0) = \frac{(\alpha N_0 T_0)^n \prod_{i=1}^{n}(N_0 - i)^\alpha}{\prod_{i=1}^{n}[(t_i(N_0 - i) + N_0 T_0)(N_0 - i)]^{\alpha+1}}.$$

Punând condiţiile de maxim funcţiei de logverosimilitate aposteriorică $\log L$ obţinem următorul sistem de ecuţii in α, N_0, T_0

$$\alpha = \frac{n}{\sum_{i=1}^{n} \log(t_i(N_0 - i) + N_0 T_0 \alpha) + \sum_{i=1}^{n} N_0 T_0 (t_i(N_0 - i) + N_0 T_0 \alpha)^{-1}}$$

$$N_0 = \frac{n}{(\alpha + 1)\sum_{i=1}^{n}(t_i + T_0\alpha)(t_i(N_0 - i) + N_0 T_0 \alpha)^{-1} - \sum_{i=1}^{n}(N_0 - i)^{-1}} \tag{5.15}$$

$$T_0 = \frac{n}{(\alpha + 1)\sum_{i=1}^{n} N_0 \alpha (t_i(N_0 - i) + N_0 T_0 \alpha)^{-1}}.$$

Observăm că acest sistem este de forma

$$\alpha = F_1(\alpha, N_0, T_0); N_0 = F_2(\alpha, N_0, T_0); T_0 = F_3(\alpha, N_0, T_0)$$

care este asemănător cu (5.11') şi prin urmare poate fi rezolvat numeric prin algoritmul descris mai inainte.

5.3 Versiunea bayessiană a modelului Jelinski-Moranda

Modelul J-M este cel cunoscut. Se presupune că timpul de funcţionare dintre căderi t_i are o repartiţie exponenţială de parametru $\lambda_i = \Phi(N - i + 1)$ unde Φ este intensitatea de apariţie a căderii (erorii) pentru ce-a de-a $i-$a cădere şi N este numărul iniţial de erori din soft. Variabilele t_i sunt stochastic independente. Versiunea bayessiană presupune [9,13,18] că N are o repartiţie discretă

$$P(N = k) = \pi_k,\ k = 0, 1, 2, \ldots \quad \sum_{k \geq 0} \pi_k = 1 \tag{5.16}$$

şi Φ are repartiţia $Gamma(0, a, b)$ adică are densitatea de repartiţie

$$g(\Phi) = \frac{a^b}{\Gamma(b)} \Phi^{b-1} e^{-a\Phi}, \quad \Phi \geq 0. \tag{5.16}$$

Funcţia de verosimilitate apriorică este

$$f(\bar{t}|N, \Phi) = \prod_{i=1}^{n} [\Phi(N - i + 1))] e^{-\Phi \sum_{i=1}^{n} (N-i+1)t_i}.$$

Densitatea de repartiţie aposteriorică a lui (N, Φ) când se dă $\bar{t}$ este

$$g(N = k, \Phi|\bar{t}) =$$

$$= \frac{\Phi^n[\prod_{i=1}^n (k-i+1)].e^{-\Phi\sum_{i=1}^n (k-i+1)t_i}\Phi^{b-1}e^{-a\Phi}\pi_k}{\sum_{l=1}^n \pi_l \int_0^\infty (l-i+1).\Phi^{n+b-1}.e^{-\Phi(a+\sum_i (l-i+1)t_i}d\Phi}, k \geq n$$

care se mai scrie

$$g(N=k,\Phi|\bar{t}) = \frac{\Phi^{b+n-1}e^{-\Phi(a+T_{n,k})}}{\Gamma(b+n)\sum_{i=1}^\infty \frac{j!}{(j-n)!}\pi_j.(a+T_{n,j})^{-b-n}} \tag{5.17}$$

unde

$$T_{n,k} = \sum_{i=1}^n (k-i+1)t_i, \quad k \geq n. \tag{5.17'}$$

Repartiţia aposteriorică a lui Φ când $N = k = const$ şi se dă $\bar{t}$ este

$$Gamma(0, a', b'), \quad a' = a + T_{n,k}, \quad b' = b + n \tag{5.18}$$

adică densitatea sa aposteriorică de repartiţie este

$$g(\Phi|\bar{t}) = \frac{\Phi^{b+n-1}e^{-\Phi(a+T_{n,k}}}{\Gamma(b+n)[a+T_{n,k}]^{-(b+n)}}. \tag{5.18'}$$

Repartiţia aposteriorică marginală a lui N (când se dă $\bar{t}$) este

$$P(N=k|\bar{t}) = \frac{\frac{k!}{(k-n)!}(a+T_{n,k})^{-b-n}\pi_k}{\sum_{j=n}^\infty \frac{j!}{(j-n)!}(a+T_{n,k})^{-b-n}\pi_j}. \tag{5.19}$$

Din (5.19) rezultă că se poate calcula estimaţia bayessiană $\hat{N}$ a lui N când funcţia de pierdere este pătratică şi anume

$$\hat{N} = \sum_{k=0}^\infty kP(N=k|\bar{t}) \tag{5.19'}$$

care se poate calcula numeric, aproximând seriile prin sume finite cu suficient de mulţi termeni.

Un caz special se obţine când N este variabila aleatoare constantă cu valoarea $N = k_0$ cunoscută şi Φ are o repartiţie apriorică $Gamma(0, a, b)$. In acest caz repartiţia aposteriorică a lui Φ când se dă $\bar{t}$ este de asemenea $Gamma(0, a', b'), a' = a + T_{n,k_0}, b' = b + n$.

• **Un alt caz special** se obţine când $\Phi = const = \Phi_0$ şi N are repartiţia apriorică $Poisson(\lambda)$. De aici se deduce asemănător formulei (5.17) că funcţia de verosimilitate a lui Φ când se dă $\bar{t}$ este forma

$$L(N = k|\bar{t}) = q\frac{k!}{(k-n)!}\Phi_0^n e^{-\Phi_0 T_{n,k}}, q = const,\ k \geq n.$$

Folosind formula lui Bayes din § 5.1 deducem că repartiţia aposteriorică a lui N este de forma

$$P(N = k|\bar{t}) = \frac{\frac{\lambda^k}{(k-n)!}e^{-\Phi_0 T_{n,k}}}{\sum_{l=1}^{\infty}\frac{\lambda^l}{(l-n)!}e^{-\Phi_0 T_{n,l}}}.$$

Dacă acum folosim relaţiile

$$T_{n,l} = lT_n - T_n - \sum_{i=1}^{n} iT_i, \quad T_n = \sum_{i=1}^{n} t_i$$

deducem expresia finală a repartiţiei aposteriorice a lui N când se dă $\bar{t}$

$$P(N = k|\bar{t}) = \frac{[\lambda e^{-\Phi_0 \sum_{i=1}^{n} t_i}]^{k-n}}{(k-n)!} \exp\{-\lambda e^{-\Phi_0 \sum_{i=1}^{n} t^i}\},\ k \geq n. \qquad (5.20)$$

Se observă că repartiţia aposteriorică a lui $N - n$ este o repartiţie Poisson de medie

$$\gamma(t_1, t_2, ..., t_n) = \lambda e^{-\Phi_0 \sum_{i=1}^{n} t_i}$$

adică estimaţia bayessiană a lui N este

$$\hat{N} = n + \gamma(t_1, t_2, ..., t_n). \qquad (5.20')$$

Deci (5.20') este estimaţia numărului iniţial de erori din soft considerat in modelui clasic al lui Jelinski-Moranda.

• **Alte versiuni ale modelului J-M.** Presupunem [18] că Φ are o repartiţie apriorică $Gamma(0, d, c)$, N este $Poisson(\lambda)$ şi t_i este exponenţială de parametru $\lambda - (i-1)\Phi$. In acest caz dându-se λ şi Φ funcţia de verosimilitate a lui $\bar{t}$ este

$$L(\bar{t}|\lambda, \Phi) = \frac{(\lambda\Phi)^n . e^{-\Phi T_{n,n}} . \exp(-\lambda(1 - e^{-\Phi t}))}{n!}, \quad t = \sum_{i=1}^{n} t_i. \qquad (5.21)$$

Presupunând că $\Phi t >> 1$ rezultă că (5.21) devine

$$L(\bar{t}|\lambda,\Phi) = \frac{(\lambda\Phi)^n \exp(-\Phi T_{n,n} - \lambda)}{n!}. \tag{5.21'}$$

Acum dacă luăm ca repartiţie apriorică pentru λ, repartiţia $Gamma(0,b,a)$, atunci se poate deduce că repartiţia aposteriorică a lui λ când se dă $\bar{t}$ este $Gamma(0,b',a')$ unde $b' = b+1, a' = a+n$ şi repartiţia aposteriorică a lui Φ când se dă $\bar{t}$ este $Gamma(0,d',c')$ cu parametri $c' = c+n, d' = d+T_{n,n}$ adică

$$p(\Phi|\bar{t}) = \frac{(d+T_{n,n})^{c+n}}{\Gamma(c+n)}\Phi^{c+n-1}e^{-\Phi(d+T_{n,n})}. \tag{5.22}$$

Acum dacă notăm $\overline{N} = N-n$ numărul de erori rămase (ne detectate), atunci, în ipotezele anterioare, după calcule, se obţine repartiţia aposteriorică a lui $\overline{N}$ când se dă $\bar{t}$ (şi când repartiţiile aposteriorice ale lui λ şi Φ sunt cele precizate anterior) de forma

$$p(\overline{N}|t) = K\frac{\Gamma(a'+N)}{\Gamma(a')\overline{N}(b')^{\overline{N}}}\left(\frac{d'}{d'+\overline{N}t}\right)^{c'} \tag{5.23}$$

unde K este o constantă de normare.

Se poate obţine o aproximare numerică a estimaţiei bayessiene a lui $\overline{N}$ şi anume o aproximare cu o sumă finită a seriei

$$\hat{\overline{N}} = \sum_{i=1}^{\infty} \overline{N}.p(\overline{N}|\bar{t}). \tag{5.23'}$$

• **Un model depinzând de probabilitatea de a avea eroare.** Presupunem că fiabilitatea depinde de probabilitatea p de a avea eroare. iar dacă există erori în soft atunci presupunem că intervalul de timp dintre apariţia a două erori consecutive este exponenţial de parametru λ. Atunci fiabilitatea este

$$R(t \mid \lambda,p) = (1-p) + pe^{-\lambda t}. \tag{5.24}$$

De aici rezultă că dacă T este perioada pe care s-a realizat testarea şi dacă selec tia (notată ca mai sus) este $\bar{t} = t_1, t_2, ..., t_n$ atunci verosimilitatea apriorică este

$$f(\bar{t} \mid \lambda,p) = \begin{cases} (1-p) + pe^{-lambdat}, & n = 0 \\ \frac{(\lambda T)^n}{n!}e^{-\lambda T} & n > 0, T = \sum_i t_i, \end{cases} \tag{5.24'}$$

unde pentru $n > 0$ se ia $p = 1$ deoarece în acest caz *sigur* s-au detectat erori. Deci cazul interesant este când $n > 0$, când presupunem că $p = 1$ şi că repartiţia apriori a lui λ este $Gamma(0, a, b)$ adică

$$h(\lambda) = \frac{a^b}{\Gamma(b)}\lambda^{b-1}e^{-a\lambda}. \tag{5.25}$$

După calcule simple se deduce că verosimilitatea aposteriori a lui λ este

$$h(\lambda \mid \bar{t}) = \frac{(a+T)^{n+b}}{\Gamma(a+T)}\lambda^{n+b-1}e^{-(a+T)} \tag{5.25'}$$

adică media şi dispersia luiλ sunt respectiv

$$E(\lambda) = \frac{b+n}{a+T}, \quad Var(\lambda) = \frac{b+n}{(a+T)^2}. \tag{5.25''}$$

Dacă $n = 0$ să presupunem că repartiţia lui p este $Beta(c+1, d+1)$ adică are densitatea apriori

$$g(p) = \frac{1}{B(c+1, d+1)}p^c(1-p)^d,\ p \in [0,1], c+1 > 0,\ d+1 > 0. \tag{5.26}$$

Atunci funcţia de verosimilitate aposteriori şi media aposteriori sunt respectiv

$$g(p \mid T) = \frac{(c+d+2)\Gamma(c+d+2)[p^c(1-p)^{d+1} + p^{c+1}e^{-\lambda T}]}{\Gamma(c+1)\Gamma(d+1)[d+1+(c+1)e^{-\lambda T}]} \tag{5.26'}$$

$$E[p \mid T] = \frac{c+1}{c+d+3}\left[1 - \frac{e^{-\lambda T}}{d+1+(c+1)e^{-\lambda t}}\right].$$

Pentru estimarea parametrilor a şi b când $n = 0$ se maximizează funcţia de verosimilitate (5.25') iar pentru estimarea parametrilor c şi d (când $n > 0$) se maximizează funcţia de verosimilitate (5.26'). Calculul efectiv al acestor estimaţii se realizează cu metode numerice.

5.4 Un model bayessian ce foloseşte repartiţia geometrică

Presupunem [35] că R_i este fiabilitatea softului în *faza a $i-a$ de testare.* (O fază de testare constă din mai multe testări individuale, numite *cazuri de testare*). Fie X_i numărul de cazuri de testare în faza a $i-$a de testare (*depanare*) până când apare o cădere (eroare de program). Deci X_i este o variabilă aleatoare discretă care presupunem că are o repartiţie geometrică de parametru R_i adică

$$P(X_i = k) = R_i^{k-1}(1 - R_i),\ k \geq 1. \tag{5.27}$$

Să presupunem în plus că R_i are o repartiţie apriorică de tip $Beta(a, b)$, adică are densitarea de repartiţie

$$f(R_i) = \frac{R_i^{a-1}(1 - R_i)^{b-1}}{B(a, b)}, \quad B(a, b) = \int_0^1 x^{a-1}(1 - x)^{b-1}dx.$$

Să observăn acum că densitatea aposteriorică a lui R_1 când se dă $X_1 = x_1$ este

$$f(R_1|x_1) = \frac{R_1^{x_1+a-2}(1 - R_1)^b}{B(a + x_1 - 1, b - 1)}. \tag{5.28}$$

Densitatea aposteriorică a lui R_2 când se dă $X_1 = x_1, X_2 = x_2$ este

$$f(R_2|x_1, x_2) = \frac{R_2^{x_1+x_2+a-3}(1 - R_2)^{b+1}}{Beta(x_1 + x_2 + a - 3, b + 2)}. \tag{5,28'}$$

In mod asemănător se deduce că densitatea aposteriorică a lui R_i când se dau $X_1 = x_1, X_2 = x_2, ..., X_i = x_i$ este

$$Beta(c_i, d_i), \quad c_i = \sum_{j=1}^{i}(x_j - 1) + a, \quad d_i = b + i. \tag{5.28"}$$

Ultima formulă ne conduce la estimaţia bayessiană a funcţiei R_i cu funcţia de pierdere pătratică; aceasta este media repartiţiei $Beta(c_i, d_i)$ definită de (5.28") adică

$$\hat{R}_i = E(R_i|x_1, x_2, ..., x_i) = \frac{c_i}{c_i + d_i}. \tag{5.29}$$

Acest model prezintă avantajul că estimează direct fiabilitatea R_i după i faze de testare.

6 Modele Statice

Prin termenul de *modele statice* vom desemna modele în care timpul nu intervine în mod explicit şi nu are niciun rol în determinarea caracteristicilor de fiabilitaate ale softului. In aceeaşi categorie de modele includem modelele care determină caracteristicile de fiabilitate pe baza complexităţii programelor, complexitate măsurată cu ajutorul *metricilor de complexitate.*

6.1 Modele frecvenţiale

Presupunem că softul are aceeaşi comportare la orice moment de timp şi la orice execuţie pe calculator, indiferent de valorile datelor de intrare sau de particularităţile hard ale sistemului de calcul. Atunci să considerăm ca de obicei variabila de stare

$$X = \begin{cases} 1 & daca\ programul\ functioneaza \\ 0 & altfel \end{cases} \tag{6.1}$$

care este o variabilă aleatoare ce nu depinde de timp, pentru care notăm

$$p = P(X = 0),\ R = 1 - p = P(X = 1). \tag{6.2}$$

In formula precedentă R este *fiabilitatea programului*, adică probabilitatea ca la o execuţie oarecare, programul să funcţioneze corect.

Se presupune deci că softul are o *comportare staţionară,* adică se comportă la fel la orice moment de timp, dacă datele sale de intrare se găsesc intr-un acelaş *domeniu.*

• **Modelul lui Nelson** ia în considerare numai rezultatele a n execuţii ale programului, presupunându-se că acesta nu este influenţat în execuţie de datele de intrare particulare luate la intâmplare sau de momentul de timp la care are loc execuţia (rularea) programului. Dacă în cele n rulări au avut loc α căderi, atunci o estimaţie a fiabilităţii este

$$\hat{R} = \hat{R}_n = 1 - \frac{\alpha}{n}. \tag{6.3}$$

Acest procedeu de estimare a fiabilităţii programului este cunoscut sub numele de *modelul lui Nelson* [25,35]. Se observă că $\hat{R}$ este o estimaţie nedeplasată şi consistentă a lui R, adică,

$$E(\hat{R}_n) = R,\ \lim_{n\to\infty} \hat{R}_n = R,\ in\ probabilitate. \tag{6.3'}$$

Să presupunem că domeniul datelor de intrare E este divizat în M sumbulţimi disjuncte, adică

$$E = \{E_1, E_2, ..., E_M\} \tag{6.4}$$

şi fie p_i=probabilitatea ca la o testare, datele de intrare să fie selectate din E_i. Atunci probabilitatea ca la o execuţie programul să funcţioneze este

$$R = \sum_{i=1}^{M} p_i R_i \tag{6.4'}$$

unde

$$R_i = P(X_i = 1), X_i = \begin{cases} 1 & daca\ sistemul\ functioneaza \\ 0 & altfel. \end{cases} \tag{6.4"}$$

Formula (6.4') defineşte fiabilitatea (statică) R în acest caz. Dacă $p_i, 1 \leq i \leq M$ sunt cunoscute, atunci R_i se pot estima cu modelul lui Nelson, iar R se poate calcula cu (6.4'). Astfel, dacă s-au făcut n_i testări cu date din E_i şi pentru datele din E_i s-au obţinut α_i căderi, atunci fiabilitatea programului se estimează astfel

$$\hat{R} = 1 - \sum_{i=1}^{M} \frac{\alpha_i}{n_i} p_i. \tag{6.5}$$

In general insă, p_i nu sunt cunoscute. Dacă mulţimile E_i sunt finite şi cunoscute, atunci notând $N_i = Card(E_i), N = \sum_{i=1}^{M} N_i$,rezultă că p_i se estimează cu $\hat{p}_i = N_i/N$.Dacă însă E_i sunt mulţimi infinite dar mărginite din $\mathbf{R}^k$, atunci p_i se pot estima cu ajutorul simulării astfel:

- Se generează N, un număr suficient de mare de puncte uniform distribuite în $E = \{E_1, E_2, ..., E_M\}$;(Acest lucru este posibil deoarece E_i şi E sunt mărginite);
- Se notează cu N_i numărul punctelor ce se află în E_i;
- Probabilităţile p_i se estimează cu formula $\hat{p}_i = \frac{N_i}{N}$.

Unele modele empirice se bazează pe observaţii privind comportarea unui program pe parcursul a mai multe testări independente. In acest caz să notăm $p(i)$ fiabilitatea programului la testarea a $i-$a. Se pot presupune următoarele forme particulare ale ale funcţiei $p(i)$, $i = 1, 2, ...$ şi anume

Forma liniară

$$p(i) = a + bi,\ a > 0,\ b > 0; \tag{6.6}$$

Forma invers liniară

$$p(i) = \beta - \frac{\alpha}{i},\ \beta = p(\infty),\ \alpha > 0; \tag{6.6'}$$

Forma exponenţială

$$p(i) = ab^i,\ a > 0,\ b > 0. \tag{6.6"}$$

Dacă la testarea i–a se fac n_i rulări ale programului şi se obţin d_i căderi, atunci p_i se estimează cu formula lui Nelson, adică $\hat{p}_i = 1 - d_i/n_i$. Cu observaţiile $(\hat{p}_i, i), i = 1, 2, ...$ (presupuse independente) se pot estima parametrii din formulele (6.6)-(6.6"), folosind de exemplu metoda celor mai mici pătrate. Estimarea parametrilor a şi b din (6,6") se face liniarizând în prealabil funcţia respectivă, adică $y(i) = \log a + i.b$, $y(i) = \log p(i)$. Se vor estima astfel prin metoda celor mai mici patrate parametrii $\alpha = \log a$ şi b; estimatia lui a este deci $a = e^{\alpha}$.

6.2 Modele "captură recaptură"

Una din aplicaţiile statisticii matematice rezolvă o problemă de tipul următor: *să se determine numărul N de peşti dintr-un lac?* Probleme similare sunt: *să se determine numărul N de insecte dintr-o pădure?* sau *să se determine numărul N de albine dintr-o prisacă?*,etc. Desigur, în fiecare caz se presupune că *teritoriile* sunt *inchise* adică nu comunică cu alte teritorii. (De ex. lacul este *inchis* şi nu comunică cu alte lacuri, sau terenul pe care se estimează numărul de insecte este deasemenea inchis).

Soluţia fiecărei probleme se poate baza pe *observaţii*, care de exemplu, pentru problema referitoare la peşti se bazează pe capturarea unor peşti, dar nu este posibilă (şi nici interesantă) din punct de vedere practic capturarea tuturor peştilor, pentru a determina numărul exact al lor. Numărul N de peşti se va *estima* conform următorului procedeu statistic:

- se capturează un număr de n peşti care se marchează într-un fel (de ex. se vopsesc) şi fără să fie sacrificaţi (sau omorâţi!) sunt din nou eliberaţi in lac. Deci, acum în lac există două feluri de peşti: peşti marcaţi şi peşti ne marcaţi; să notăm

$$p = \frac{n}{N} \tag{6.7}$$

probabilitatea ca un peşte din lac să fie marcat;

- se capturează din nou un număr $\nu << N$ de peşti din acelaş lac; dacă α este numărul de peşti marcaţi din această cea de-a doua captură, atunci o estimare a lui p este

$$\hat{p} = \frac{\alpha}{\nu}. \tag{6.7'}$$

Acum în (6.7) se cunosc $n, \hat{p}$ şi deci se poate estima N ca $\hat{N} = n/\hat{p}$.

Procedeul se aplică numai în cazul când N este foarte mare, astfel încât la cea de-a doua captură să putem considera *extragerile fără întoarcere,* astfel ca probabilitatea p să nu se modifice. Dacă şi ν este mare atunci putem determina şi un interval de încredere pentru N care este de forma

$$[\hat{N}_1, \hat{N}_2] \quad \hat{N}_1 = \frac{N}{\hat{p}_2}, \quad \hat{N}_2 = \frac{N}{\hat{p}_2}$$

$$[\hat{p}_1, \hat{p}_2], \hat{p}_1 = \hat{p} - z_\delta \frac{\hat{\sigma}}{\sqrt{\nu}}, \hat{p}_2 = \hat{p} + z_\delta \frac{\hat{\sigma}}{\sqrt{\nu}} \tag{6.8}$$

unde

$$\hat{\sigma} = \hat{p}(1 - \hat{p}), \quad \frac{1}{\sqrt{2\pi}} \int_{-z_\delta}^{z_\delta} e^{-\frac{u^2}{2}} du = \delta. \tag{6.8'}$$

In formula precedentă δ este *coeficientul de încredere* sau de *toleranţă,* adică o probabilitate dată, apropiată de 1, dar mai mică decât 1. Pentru determinarea intervalului de încredere $[\hat{p}_1, \hat{p}_2]$, având dat coeficientul de încredere δ apropiat de 1 (de ex. $\delta = 0.95$), s-a folosit teorema limită centrală, adică faptul că variabila aleatoare $Z = (\hat{p} - p)/(\hat{\sigma}/\sqrt{\nu})$ are, pentru $\nu-$mare, o repartiţie normală $N(0, 1)$.

Procedeul statistic descris anterior, pentru estimarea numărului N de peşti din lac, este de fapt un procedeu general pentru rezolvarea prolemelor de acest tip şi este cunoscut sub numele de *procedeul captură-recaptură.*

Soluţia problemei analizate anterior, bazată pe procedeul captură-recaptură, se poate transpune şi în cazul estimării numărului N_0 de erori din sistemul soft. Astfel, după un număr de testări din care s-au detectat multe erori, se introduc din nou cele n erori detectate. Să numim aceste erori *însămânţate.* Acum, probabilitatea de a detecta într-o testare ulterioară a programului o eroare însămânţată, este dată de (6.7) cu $N = N_0$. Dacă acum, analog procedeului de mai sus, efectuăm alte ν noi rulări ale softului (adică noi testări),

din care detectăm α erori sau eşecuri în execuţia softului, atunci p se estimează din nou cu $\hat{p}$ dat de (6.7'). Deci N_0 se estimează acum cu formula $\hat{N}_0 = n/\hat{p}$ si un interval de încredere pentru N_0 se obţine tot cu relaţiile (6.8) şi (6.8').

Remarcă. Dacă N_0 nu este suficient de mare şi ν nu este suficient de mare, atunci $P(\alpha = i)$ se calculează pe baza repartiţiei hipergeometrice, adică

$$P(\alpha = i) = \frac{C_n^{\alpha} C_{N_0}^{\nu-\alpha}}{C_{N_0+n}^{\nu}}$$

ultima probabilitate fiind funcţia de verosimilitate cu care se poate estima N_0 prin metoda verosimilităţii maxime. Dacă $\hat{N}_0$ este estimaţia de verosimilitate, atunci intervalul de încredere pentru N_0 se poate determina cu ajutorul repartiţiei discrete a lui $\hat{N}_0$, problemă de care nu ne ocupăm aici, ea putând fi rezolvată uşor pe baza unui algoritm.

6.2.1 Alte tipuri de modele captură-recaptură.

•**Un model alternativ de tip "captură recaptură"** se obţine când testarea experimentală se realizează cu două echipe de programatori. Erorile depistate de prima echipă (în număr de n) sunt *însămânţate* în program şi se face apoi testarea programului de către a doua echipă. Dacă ν este numărul de erori detectate de cea de-a doua echipă şi dintre acestea α sunt erori însămânţate, atunci cu aceleaşi notaţii, numărul N_0 de erori iniţiale ale programului se poate estima cu aceeaşi metodologie ca în caazul modelului "captură recaptură".

• **Alt model alternativ de tip "captură recaptură"** se obţine când testarea se realizează în paralel cu două echipe astfel: se fac testări de către cele două echipe obţinându-se respectiv n_1, n_2 erori din care n $(n < n_1, n_2)$ sunt erori comune; deci probabilitatea unei erori comune la o testare este $p = n/N_0$; într-o nouă fază de testare se fac de către cele două echipe ν_1 respectiv ν_2 testări şi se obţin α erori comune. Atunci p se estimează cu formula

$$\hat{p} = \frac{\alpha}{\nu}, \nu = \nu_1 + \nu_2 \qquad (6.8")$$

de unde procedeul continuă ca mai sus, adică

$$\hat{N}_0 = \frac{n}{\hat{p}}.$$

Intervalul de încredere pentru N_0 se determină tot cu formulele (6.18), (6.18").

6.2.2 Modele generalizate de tip captură-recaptură.

Deoarece în general erorile din soft nu au aceeaşi probabilitate de apariţie iar membrii echipelor (*inspectorii*) care testează softul, sau inspectează documentaţiile (în care de asemenea pot exista erori), nu au aceleaşi performanţe, utilizrea procedeului captură-recaptură trebuie adaptată în mod corespunzător. In acest caz general datele obţinute pe o perioadă mică de timp se presupun de forma

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ x_{21} & x_{22} & \dots & x_{2k} \\ . & . & \dots & . \\ x_{D1} & x_{D2} & \dots & x_{Dk} \end{pmatrix}$$

unde $x_{ij} = 1$ dacă şi numai dacă inspectorul i a detectat eroarea $j, 1 \leq i \leq k, 1 \leq i \leq D$. Dacă testarea se efectuează pe perioade mari de timp atunci datele obţinute constituie o matrice de forma

$$\mathbf{N} = \begin{pmatrix} N_{11} & N_{12} & \dots & N_{1k} \\ N_{21} & N_{22} & \dots & N_{2k} \\ . & . & \dots & . \\ N_{k1} & N_{k2} & \dots & N_{Dk} \end{pmatrix}$$

unde N_{ij} este numărul de erori de tipul j detectate de inspectorul i.

Din matricea de date X se poate determina:

- pentru fiecare inspector i, numărul total de erori detectate de acesta $N_{i.} = \sum_{i=1}^{D} N_{ij}$;
- pentru fiecare tip de eroare j, numărul de inspectori care detectează acea eroare $N_{.j} = \sum_{i=1}^{k} N_{ij}$;
- numărul total de erori detectate $N = \sum_{i=1}^{k} \sum_{j=1}^{D} N_{ij} = \sum_i Ni. = \sum_j N.j$.

Datele precedente se obţin în faza "captură". Dacă erorile detectate sunt "însămânţate" atunci în faza recaptură, când erorile X_{ij} obţinute in prima fază sunt *marcate*, se obţin datele

$$\nu = ||\nu_{ij}||$$

$$\mathbf{A} = ||a_{ij}||, \quad a_{ij} \leq \nu_{ij}$$

unde ν_{ij} = numărul total de erori de tipul j detectate de inspectorul i în faza recaptură iar a_{ij} = numărul de erori *recapturate* de tipul j detectate de inspectorul i.

Să notăm acum $N_{(0)j}$= numărul de erori de tipul j existente in soft, N_0 = umărul total inițial de erori din soft şi fie $p_{.j}$= probabilitatea unei erori de tipul j, $p_{i.}$= probabilitatea detectării unei erori de către inspectorul i şi fie p_{ij}=probabilitatea ca o eroare de tipul j să fie detectată de inspectorul i. Facem ipoteza naturală că $p_{ij} = p_{i.}p_{.j}$ Ne interesează estimaţii pentru $N_{(0)j}$ şi pentru N_0. Introducând notaţii asemănătoare celor definite pentru matricea **N** şi pentru matricile ν şi **A** se constată că estimaţiile probabilităţilor introduse sunt

$$\hat{p}_{ij} = \frac{a_{ij}}{\nu_{ij}}, \quad \hat{p}_{i.} = \frac{a_{i.}}{\nu_{i.}}, \quad \hat{p}_{.j} = \frac{a_{.j}}{\nu_{.j}}$$

In concluzie, în final se obţin estimaţiile

$$\hat{N}_{(0)j} = \frac{N_{.j}}{\hat{p}_{.j}}, \quad \hat{N}_0 = \sum_{j=1}^{D} N_{.j}.$$

Cu ajutorul estimaţiilor probabilităţilor se poate verifica (utilizând un test χ^2) şi ipoteza $H : p_{ij} = p_{i.}p_{.j}$. De asemenea N_0 se poate estima şi cu formula

$$\hat{N}^*{}_0 = \sum_{i=1}^{k} \hat{N}^*{}_{(0)i.}, \quad \hat{N}^*{}_{(0)i.} = \frac{N_{i.}}{\hat{p}_{i.}}.$$

Cele două estimaţii ale lui N_0 permit estimarea erorii relative

$$RE(N_0) = \frac{\hat{N}_0 - \hat{N}^*{}_0}{\hat{N}_0}.$$

Eroarea relativă $RE(N_0)$ ne dă informaţii asupra eterogenităţii echipei de inspectori.

6.3 Modele bazate pe metrici de complexitate

In general există o strânsă legătură între numărul de erori din program şi complexitatea programului exprimată prin numărul de instrucţiuni şi prin natura acestora (număr de linii, număr de *căi de testare*, număr de cicluri sau blocuri predicative, etc.).

Halstead [14,18] propune o serie de *atribute* ale programului care sunt determinate dintr-un număr de *metrici* ce derivă din structura unui program şi din implementarea acestuia într-un limbaj de programare. Atributele unui

program includ mărimea, efortul mental de a crea programul. timpul necesar creierii programului şi numărul de elemente esenţiale ce compun structura programului. Structura programului poate fi considerată în sensul teoremei Böhm-Jaccopini (programul este reprezentat sub formă de schemă logică) sau în sensul *modular-ierarhic* al teoremei lui Jackson (programul este reprezentat printr-o diagramă modulară de structură, numită *diagramă de structură nested logic*). Aici vom prezenta câteva modele *empirice* bazate pe aşa zisele *metrici de complexitate* ale softului. Ne vom referi la acele metrici care au legătură cu fiabilitatea programelor.

6.3.1 Model bazat pe metrica de tip Halstead.

Metricile de complexitate sunt în special legate de *mărimea* softului şi presupun că în funcţie de valoarea metricei se poate estima numărul de erori din soft. Mai întâi vom prezenta câteva *caracteristici metrice* ale unui program analizând construcţia *structurată* a acestuia.

Se ştie că structurile (sau blocurile) *primitive* ale unui program în sensul *Böhm-Jaccopini* sunt:

- Secvenţa Π: **begin** $S_1, S_2, ..., S_n$ **end**;

- selecţile, de patru tipuri:

a) Φ_1::= **if** P **then** S; (adică selecţia "if-then");

b) $\Delta = \Phi_2$::= **if** P **then** S_1 **else** S_2; (adică selecţia "if-then-else");

c) Φ_3::=**if** P_1 **then** S_1 **elseif** P_2 **then** S_2 ... **elseif** P_n **then** S_n; (adică selecţia multiplă în "cascadă");

d) Φ_4::=**case** I **for** $I_1 : S_1, I_2 : S_2, ..., I_n : S_n$; (adică selecţia multiplă de tip "case", cu predicat polivalent);

- iteraţiile, de trei tipuri

e) Ω_1::= **for** I **do** S; (adică ciclul "for);

f) $\Omega = \Omega_2$::= **while** P **do** S; (ciclul "while");

g) Ω_3::= **repeat** S **until** P; (ciclul "repeat-until"). Să observăm că formal, toate structurile de mai sus se pot transforma în programe echivalente care se pot exprima recursiv numai in termeni de Π, Δ, Ω.

In structurile de program de mai sus S_i reprezintă instrucţiuni simple, blocuri funcţionale (de forma $S : X \to X'$, care transformă o mulţime de date X într-o mulţime de date X'), sau alte blocuri primitive; I reprezintă o variabilă discreta cu o mulţime finită de valori (deobicei I este întreg), iar $P, P_i, 1 \leq i \leq n$ sunt predicate simple (cu două valori:*true* sau *false*).

Un program care constă din *compunerea* recursivă de structuri primitive de tipul celor de mai sus se numeşte program structurat. O teoremă a lui Böhm şi Jaccopini spune că *orice program* (algoritm) poate fi transformat într-un program structurat echivalent. De fapt teorema spune că orice program se poate reprezenta ca o combinaţie recursivă, finită a trei din structurile de mai sus şi anume Π, Δ, Ω, dacă se adaugă la nevoie noi blocuri predicative ω şi noi blocuri funcţionale de forma $T ::= \omega := true$, şi $F ::= \omega := false$.

Să observăm că un program structurat, când este implementat într-un limbaj, nu va conţine instrucţiunea *goto* necondiţionat.

Este deci suficient să definim măsuri de complexitate numai pentru structurile de program de mai sus, deoarece acestea, prin combinări recursive între ele vor conduce la calculul metricii oricărui program structurat.

Se presupune că o măsură M a complexităţii algoritmilor reprezentaţi de srtucturile primitive menţionate mai sus satisface următoarele axiome:

1. $M(\Pi) = \sum_{i=1}^{n} M(S_i)$;
2. $2M(S) \geq M(\Phi_1) > M(S)$;
3. $2(M(S_1) + M(S_2)) \geq M(\Phi_2) > M(S_1) + M(S_2)$;
4. $2\sum_{i=1}^{n} M(S_i) \geq M(\Phi_3) > \sum_{i=1}^{n} M(S_i)$;
5. $M(\Phi_4) > \sum_{i=1}^{n} M(S_i)$;
6. $M(\Omega_1) > M(S)$;
7. $2M(S) \geq M(\Omega_2) > M(S)$;
8. $2M(S) \geq M(\Omega_3) > M(S)$;

In axiomele de mai sus se recunosc (măcar parţial) axiomele conceptului clasic de *măsură*.

O *metrică* de complexitate M este o aplicaţie $M : \mathcal{P} \to \mathbf{R}^+$ care se presupune că satisface axiomele de mai sus, unde $\mathcal{P}$ este mulţimea programelor structurate.

• **Teoria lui Halstead.** Halstead, introduce următoarele caracteristici (masurabile) ale unui program, deduse din considerente plausibile:

Numărul n_1 de *operatori distincţi* în program (care sunt cei ce definesc strucrurile primitive), $n_1 = const$;

Numărul n_2^* de parametri input/output ai programului;

Numărul n_2 de operanzi distincţi din program, $n_2 = bn_2^*$, unde b este o constantă.

Numerele N_1, N_2 ce reprezintă numărul total de operatori, respectiv operanzi, din program.

Numărul $n = n_1 + n_2$ reprezintă (conform lui Halstead) *dicţionarul programului;*

Lungimea programului este definită de formula

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2 \tag{6.9}$$

Volumul programului este (cf.Halstead)

$$V = N \log_2 n = (n_1 \log_2 n_1 + n_2 \log_2 n_2) \log_2 (n_1 + n_2). \tag{6.9'}$$

In funcţie de acestea se definesc alte caracteristici (măsurabile) care pot exprima complexitatea unui program şi anume:

Nivelul programului este

$$L = \frac{2n_2}{n_1 N_2}; \tag{6.10}$$

Dificultatea programului este

$$D = \frac{n_1 N_2}{2n_2}; \tag{6.11}$$

Nivelul limbajului (în care este implementat programul) este

$$T = \frac{VD}{18} \quad , (formula\ dedusa\ empiric) \tag{6.12};$$

Nivelul programului este

$$\lambda = \frac{V}{D^2} \tag{6.13}$$

Coeficienţii numerici din formulele de mai sus, precum şi din alte formule ce vor fi precizate mai jos, au fost deduşi pe baza analizei statistice a unor clase variate de programe complexe (compilatoare, utilitare, SGBD-uri, etc.); aceşti coeficienţi s-au estimat cu metodele *teoriei regresiei* folosind date istorice culese explorând diversele programe existente în exploatare (date *istorice*).

Ideile introduse de Halstead sunt importante pentru că identifică un număr de elemente cantitative simple care pot constitui măsurători şi *observaţii primare* pentru a legitima o *ştiinţă a software-ului.*

Caracteristicile V, D, L, T introduse anterior pot fi utilizate ca *metrici* ale unui program.

Să arătăm de exemplu că D satisface axiomele 1-8 introduse mai sus. Când într-o structură de program va interveni un bloc S vom menţiona la indicii diverselor caracteristici şi indicele s. Vom verifica pe rând axiomele pentru D.

1. $D(\Pi) = \sum_{i=1}^{n} M(S_i) = \frac{n_1 \sum_{i=1}^{n} N_{2s_i}}{2n_2}$;
2. $2D(S) = 2\frac{n_1 N_{2s}}{2n_2} \geq D(\Phi_1) = D(S)$;
3. $2(D(S_1) + D(S_2)) = \frac{2(n_1 N_{2s_1} + n_1 N_{2s_2}}{2n_2} \geq D(\Phi_2) > \frac{n_{1s_1} N_2}{2n_2} + \frac{n_{1s_2} N_2}{n_2} = D(S_1) + D(S_2)$;
4. $2\sum_{i=1}^{n} D(S_i) \geq \frac{n_1 N_2}{2n_2} > \frac{(n_{1s_1} + n_{1s_2}) N_2}{2n_2} = D(S_1) + D(S_2)$;

şi a.m.d.

In mod asemănător se pot verifica axiomele 1-8 şi pentru V.

Metrica lui Halstead (cunoscută sub acest nume) este mărimea V, care după cum se vede ţine seama atât de structura cât şi de complexitatea programului.

Halstead propune ca estimaţie a numărului de erori iniţiale din soft

$$\hat{N}_0 = \frac{V}{E_0} \qquad (6.9")$$

unde V este volumul programului propus de Halstead, iar E_0 este o constantă de proporţionalitate ce se estimează din date istorice. Desigur, modelul fiind empiric, i se pot aduce multe critici. El a rezultat din date experimentale efectuate asupra multor testări de soft, iar formulele (6.9)-(6.9") reprezintă valori *medii* sau *ordine de mărime*.

6.3.2 Alte modele bazate pe metrici

Metricile de complexitate a programelor s-au dezvoltat pornind de la faptul că orice model de dezvoltare de software trebuie să ţină seama de descrierea *produsului soft*, de *procesele* şi *evenimentele* asociate acestuia şi legat de acestea, de *atributele* pe care le capătă produsul soft implementat.

Produsele soft posedă două atribute cuantificabile importante: *mărimea şi structura* aşa cum am precizat şi mai sus. Dacă mărimea este uşor de imaginat (prin lungimea textelor sursă sau executabile de exemplu), structura este măsurată în funcţie de *atributele relaţiilor* dintre componente care adesea sunt modelate prin grafuri (vezi schemele logice!) sau prin *diagrame de structură ierarhice, arborescente* (vezi de exemplu *diagramele de structură "nested logic"*). Structura programelor executabile poate fi cuantificată

în funcţie de caracteristicile interfeţelor modulelor ce alcătuiesc programul. Pot fi cuantificate şi textele care descriu *specificaţiile* programului, mai ales în ceea ce priveşte posibilitatea înţelegerii acestora de către utilizator. Cuantificabile sunt şi *relaţiile între produse* precum şi *rata expansiunii produselor* care măsoară de exemplu relaţia între mărimea unei documentaţii de specificare şi mărimea modulelor de program sursă sau executabil corespunzătoare. In sfârşit, se poate cuantifica şi *stabilitatea* sau *instabilitatea* vis a vis de dezvoltarea sau intreţinerea produselor soft.

Procesele sunt de două tipuri:*task-uri* şi *operaţii.* Task-urile sunt obiectivele de proiectare a unui subsistem sau a unei unităţi de testare a unui modul specificat care sunt identificate în mod explicit pentru realizarea produselor. La rândul lor task-urile pot fi de două tipuri: task-uri de producţie şi task-uri de verificare (testare), care toate sunt planificate în timp. Multe metrici ale proceselor se obţin din task-uri de testare şi ele se referă la *completitudinea activităţilor de testare* sau la *eficienţa* acestei activităţi. Operaţiile sunt proceduri care se execută în cazuri neprevăzute (de exemplu ce se face când un modul cade la testare), sau în alte cazuri când nu a fost posibilă o planificare.

Evenimentele sunt inţelese în acest context ca semnalele care cauzează activarea proceselor. Cele mai multe metrici bazate pe evenimente derivă din numărarea erorilor găsite la testare sau pe cerinţele de modificare a programelor pe parcursul implementării lor. De obicei metricile bazate pe evenimente sunt incorporate în cele bazate pe produse şi procese.

Metricile ce vor fi descrise în continuare sunt extrase din diverse surse documentare (printre care sursa primară este lucrarea [18]). Ele sunt dintre metricile care s-au dovedit utile în practică, au fost validate de experienţa producătorilor de soft, sunt cele mai cunoscute deşi unele şi-au pierdut din interes, sau sunt cele care indeplinesc cerinţe importante în activitatea de realizare a produselor soft. Enumerăm pe cele mai importante.

Numărul de linii de program. Este folosit pentru a măsura module, subsisteme sau sisteme; este o metrică necesară în controlul şi normarea muncii şi pentru prognoze. Este legată de *lungimea N* a programului în sensul lui Halstead.

Numărul de cuvinte. Este o măsură statică a mărimii fizice a unui a unui document scris în limbaj natural care este fie document de specificare fie manual de utilizare.Este folosit ca metrică de control. Metrici echivalente

sunt numărul de *fraze* sau numărul de *pagini*.

Complexitatea proiectului soft. Măsoară complexitatea produsului soft în termeni de număr de module care compun proiectul şi de conexiuni dintre acestea. Este un indicator de calitate care este legat de *claritate* şi caracterizează facilitarea înţelegerii şi posibilitatea de a întreţine produsul soft.

Măsurătorile de bază care sunt necesare pentru a construi metrica de complexitate a softului sunt:

A_i= numărul de arce din descrierea modular-ierarhică a programului de la nivelul 0 la nivelul i;

N_i= numărul total de module de la nivelul 0 la nivelul i;

T_i= numărul total de arce din structura arborescentă *ideală* a programului; structura arborescentă modulară este ideală când fiecare modul este apelat de un singur modul.Din definiţie rezultă $T_i = N_i - 1$.

Din caracteristicile numerice definite anterior rezultă următoarele măsuri ale calităţii proiectării unui produs soft:

$C_i = A_i - T_i$ este măsura complexităţii absolute a nivelului i al arborescenţei modulare;

$R_i = C_i/A_i$ este măsura complexităţii relative a nivelului i al structurii modulare;

$D_i = (C_i - C_{i-1})/(A_i - A_{i-1})$ este măasura complexităţii relative a nivelului ierarhic i al structurii modulare (numită şi impuritatea relativă a arborelui).

Aceste metrici se referă la domenii de proiectare software suboptime şi deci de calitate scăzută. Totuşi analize practice au evidenţiat o legătură intre C_i şi R_i pe de-o parte şi rata erorilor din programe pe de-altă parte.

Complexitatea ciclomatică. Aceasta se bazează pe *numărul ciclomatic* introdus de McCabe [8,9,10.18]. Programul este reprezentat de o schemă logică G (un graf orientat cu un nod de intrare-start si unul final-stop); numărul ciclomatic este

$$V(G) = e - n + 2p \tag{6.10}$$

unde e este numărul de arce ale schemei logice, arcul fiind o muchie de *ramificare*, n este numărul de noduri, unde un nod este echivalent cu un bloc secvenţial din programul sursă, iar p este numărul de componente obţinute prin conectare, adică în mod normal este 1.

In programe structurate (adică programe care nu permit salturi în şi din cicluri), $V(G)$ este echivalent cu numărul de predicate plus 1, unde predicatele compuse din blocuri de forma "IF α AND β THEN" sunt considerate ca două. Acesta este echivalent cu numărul de puncte de decizie din program.

McCabe a sugerat că un program cu o metrică de valoare mare reprezintă un soft dificil de produs şi de intreţinut. Există autori care îşi exprimă rezerve faţă de utilizarea metricii definite de numărul ciclomatic al lui McCabe. El poate fi utilizat mai degrabă ca măsură a *testabilităţii* unui program.

Măsura de complexitate a lui Oviedo. Aceasta este o sumă ponderată a metricii de control a fluxului şi a metricii fluxului de date. Ea măsoară complexitatea unui program şi este de forma

$$C = aCF + bDF \tag{6.11}$$

unde CF este complexitatea de control a fluxului măsurată prin numărul de muchii din schema logică, DF este complexitatea fluxului de date măsurată prin suma complexităţilor fluxurilor de date din fiecare bloc al programului, iar a şi b sunt ponderi ce pot fi presupuse egale cu 1. Această metrică poate fi utilizată pentru a identifica modulele complexe care ar necesita o reproiectare sau testări suplimentare. In literatură se comentează că această metrică a lui Oviedo nu se ridică deasupra interesului reprezentat de metrica lui McCabe.

Importanţa sau *tăria* modulului. Această metrică reprezintă aşa zisa *coeziune* a unui modul, adică în ce măsură un modul este considerat ca având o singură funcţiune. Se pleacă de la ideea că un modul bine modularizat (sau segmentat) are de regulă o singură funcţie. Deci modulele care realizează mai multe funcţiuni sunt super-complexe şi exprimă faptul că sistemele nu au fost suficient de descompuse. Deci *tăria* unui modul este atât un indicator al calităţii descompunerii cât şi al complexităţii modulelor individuale. Unii autori recomandă ca tăria unui modul să indice cât de multe din următoarele funcţiuni realizează modulul: intrări/ieşiri; lansări în execuţie/ comenzi de control; prelucrări algoritmice. Măsura acestor funcţiuni poate deci avea valorile 1, 2 sau 3, unde valoarea 1 este considerată ca importanţă mare, valoarea 2 este o importanţă medie şi valoarea 3 este o importanţă redusă.

Se observă că această metrică se exprimă prin valori alese in mod *subiectiv* şi deci se recomandă să fie puţin utilizată; ea poate indica o modularizare necorespunzătoare a componentelor softului.

Metrici "fan-in şi fan-out". Aceste metrici contorizează interconexiunile unui modul cu alte module ale sistemului. Termenii "fan-in" şi "fan-out" sunt folosiţi în mod ambiguu în literatura despre metricile de software pentru a desemna diferite concepte ca:

(i) să descrie relaţiile de apelare dintre module ilustrate prin diagrama de structură a softului; acestea sunt numite fan-in şi fan-out structurale;

(ii) să descrie relaţiile fluxurilor de date dintre proceduri şi relaţiile dintre proceduri şi date; acestea sunt numite fan-in şi fan-out informaţionale.

Modulele sunt privite aici ca unităţi de compilare singulare.

Metricile structurale fan-in numără modulele care apelează un modul, în timp ce metricile structurale fan-out numără modulele apelate de un modul dat. Ele sunt indicatori de calitate in sensul că modulele cu valori mari ale metricii structurale sunt şi complexe şi critice din punct de vedere sistemic.

Metricile informaţionale fan-in şi fan-out se bazează pe fluxul de informaţii dintre proceduri care nu se rezumă la faptul că un modul apelează pe celălalt (numit *flux local* de informaţii), dar şi pe informaţiile bazate pe valorile returnate prin apel (numit *flux local indirect*) precum şi pe informaţiile transferate între proceduri prin structuri de date globale (numite *fluxuri globale* de informaţii). Metrica informaţională fan-in numără fluxul de date locale în procedură şi numărul structurilor de date din care procedura extrage informaţii; metrica informaţională fan-out a unei proceduri numără fluxul de date locale ce ies din procedură plus numărul de structuri de date pe care procedura îl actualizează.

Fluxul local de date apare dacă se indeplineşte una din următoarele condiţii.

- o procedură apelează o altă procedură;
- o procedură apelează o altă procedură şi apoi utilizează o valoare ce i se returnează (aceasta este considerată fan-in pentru procedura apelantă şi fan-out pentru procedura apelată);
- o procedură apelează două proceduri A şi B şi ieşirea lui A este intrare în B (aceasta este fan-out din A şi fan-in în B).

Complexitatea fan-in şi fan-out (introdusă de către Henry şi Kafura) este:

$$C_{fi-fo} = lungimea \times [fan - in \times fan - out] \qquad (6.12)$$

unde lungimea este orice metrică de "volum" (de exemplu numărul de linii ale programului, sau mărimea programului după Halstead, sau numărul ciclomatic al lui McCabe).

Metricile structurale sunt potrivite pentru produsele software care utilizează în proiectare diagrame de structură ierarhice şi sunt recomandabile pentru evaluarea acestor diagrame.

Metricile informaţionale par a avea o importanţă mai mare în probleme de proiectare. Conceptele fan-in şi fan-out pot fi extinse şi la *structuri de date* prin descrierea *structurată* a acestora.

Formula facilităţii de citire a programului. Această măsură R este dată de formula

$$R = 0.295VAR - 0.499NSL + 0.13CYCLO \tag{6.13}$$

unde VAR este lungimea medie normalizată a variabilelor textului programului, NSL este numărul de linii care conţin instrucţiuni, iar $CYCLO$ este numărul total de ramificaţii din program plus unu.

O măsură a facilităţii de citire a programului este necesară pentru intreţinerea şi extinderea acestuia. Acesta este cazul programelor care se utilizează sub diverse versiuni pe perioade îndelungate.

Indicele de mascare. Este o metrică de calitate a facilităţii de citire şi înţelegere a unui text. Caracteristicile măsurabile ale unui text produs pentru a fi citit sunt: *sen*= numărul de propoziţii; *wrd*= numărul de cuvinte; *syl*=numărul de silabe. Indicele de mascare a citirii este

$$FI = 0.4(wrd/sen + (hrd/wrd)100) \tag{6.14}$$

unde *hrd* este numărul cuvintelor de trei sau mai multe silabe.

Acest indice este mai puţin utilizat pentru a măsura calitatea documentaţiilor de software; în schimb el este utilizat pentru evaluarea textelor din reviste de anunţuri sau reclame de programe.

Rata de extindere a proiectului. Această metrică se referă la situaţiile când proiectarea softului se face într-un limbaj formal specializat de proiectare (de tipul unui pseudocod). Rata de extindere a proiectării este

$$E = LOC/DS$$

unde LOC este numărul total de linii-cod (scrise în limbajul dat) iar DS este numărul total de instructiuni din proiect. Este una din puţinele metrici care consideră relaţiile dintre produse.

Ratele de erori şi de modificări. Metricile de acest tip măsoară volumul de schimbări la care este supusă o componentă soft pe parcursul dezvoltării produsului.

Măsurătorile utilizate de acest tip de metrici sunt:

- numărul total de erori găsite într-o componentă;
- numărul total de erori de un anumit tip găsite într-o componentă;
- rata erorilor= (total erori)/(volumul componentei);
- rata erorilor pentru un anumit tip de eroare;
- volumul modificărilor componentei datorate erorilor;
- volumul modificărilor componentei datorate erorilor de un anumit tip.

Tipurile de erori ce pot fi considerate sunt:

- erori ce au fost introduse în componentă de un proces particular (de exemplu de procesul de proiectare);
- erori evidenţiate de un proces particular (de exemplu prin controale);
- erori de o natură particulară (de exemplu s-a omis ceva sau s-a introdus ceva ce este incorect);
- erori legate de un tip particular de specificaţie software (de exemplu erori de intrare/ieşire sau erori logice de specificare).

Volumul modificărilor şi erorilor, precum şi ratele acestora pot avea importanţă în estimarea costurilor de proiectare şi testare.

Subliniem din nou faptul că în formulele de mai sus, coeficienţii numerici care apar, au fost estimaţi prin metode statistice folosind date istorice.

Metricile introduse mai sus au la bază considerente practice intuitive. In subsecţiunea **6.4** (următoare) vom prezenta unele consideraţii care să justifice şi din punct de vedere formal construcţia lor.

6.3.3 Modele bazate pe metrici pentru determinarea lui N_0

Una din problemele importante ale fiabilităţii programelor este, aşa cum am subliniat mai ales în capitolele 2,3,5, estimarea numărului iniţial de erori din soft, N_0.

Unul din scopurile cunoaşterii metricilor de software, aşa cum am menţionat cu ocazia prezentării metricii de tip Halstead, este şi acela de a stabili formule pentru calculul numărului iniţial de erori N_0 în funcţie de valorile diverselor metrici. Aceste formule au fost stabilite de regulă pe baza analizei unor date statistice (istorice) culese cu ocazia realizării unor produse soft existente in exploatare.

Vom enumera în continuare câteva din modelele empirice reprezentative pentru estimarea lui N_0 folosind valorile estimate ale unor metrici.

Modelul lui Lipow care spune că dacă S este numărul de instrucţiuni executabile din programul sursă, atunci

$$N_0 = (a_1 + a_2 \log S + a_3 \log S^2)S \tag{6.16}$$

unde a_1, a_2, a_3 sunt constante ce pot fi estimate folosind date $N_{0i}, S_i, 1 \leq i \leq M$ cunoscute anterior pentru M produse soft. (Se poate de ex. folosi metoda celor mai mici pătrate, care conduce la un sistem neliniar în cei trei parametri).

Lipow propune şi modelul

$$N_0 = a + bS^c \tag{6.16'}$$

unde de asemenea a, b, c sunt parametri ce trebuie estimaţi din date experimentale. O valoare constantă [18,25,35] ($c = 4/3$) a fost găsită experimental.

Modelul lui Schneider [18,25,35] constă în estimarea lui N_0 după formula

$$N_0 = 7.6E^{2/3}S^{1/3} \tag{6.17}$$

unde E este efortul de muncă profesională exprimat în *oameni-luni*, iar S este numărul de mii de instrucţiuni sursă executabile ale softului.

O altă formulă empirică de calcul aproximativ al lui N_0 este [18]

$$N_0 = n^{2/3}\left(\frac{S}{0.047}\right)^{5/3} \tag{6.18}$$

unde n este numărul de subprograme ale softului analizat.

6.3.4 Câteva modele euristice cantitative

Aceste modele, preluate din [18], se referă la controlul procesului de dezvoltare a produsului soft în funcţie de erorile găsite pe parcursul testării şi organizării procesului de testare. Ele au ca scop estimarea performanţelor intermediare ale produsului soft.

• **Model pentru sursele de erori şi detectarea lor.** Acest model foloseşte ca măsurători (date de intrare) numărul mediu de *clase* de erori n şi numărul mediu de erori f_{ij} din fiecare clasă $j = 1, 2, ..., n$ şi pentru fiecare componentă $i = 1, 2, ..., N$ a sistemului soft descris ierarhic de cele N

componente. Se presupun date şi cele $r + 1$ nivele ale diagramei ierarhice şi se presupune că procesul testării constă din r etape de testare independente. Trebuie de asemenea cunoscută *rata de detectare a erorilor* d_{jk}, pentru fiecare etapă de testare $k = 1, 2, ..., r$.

Modelul îşi propune să estimeze F'_j=numărul total de erori rămase în clasa j după integrarea tuturor modulelor calculat cu formula

$$F'_j = F_j(1 - D_j)$$

cu

$$D_j = 1 - \prod_{k=1}^{r}(1 - d_{jk}), \quad F_j = \sum_{k=1}^{r}(\sum_{i=1}^{N} f_{ij})(1 - d_{jk})$$

unde F_j = rata de apariţie a erorii în clasa j iar D_j este rata de detectare a erorii într-o etapă de testare. Desigur, modelul poate fi aplicat cu succes numai în măsura în care se pot estima bine datele de intrare.

• **Model pentru eliminarea erorilor.** Acest model presupune că elaborarea produsului soft începe cu o primă versiune ce conţine un număr de erori iar procesul de punere la punct a programului continuă cu o altă etapă de testări, revizuiri sau verificări prin care se detectează şi elimină din erorile iniţiale.

Datele de intrare ale modelului sunt: MP=numărul de probleme majore înregistrate pe parcursul revizuirilor sau verificărilor; PTM= numărul de erori descoperite pe parcursul testărilor ($MP > PTM$).

Notând

$$\mu = \frac{MP}{PTM},$$

parametri de ieşire ai modelului sunt

$$TD = MP\frac{\mu}{\mu - 1}, \quad Q_2 = \frac{TD}{\mu^2}, \quad CRE = \frac{\mu^2 - 1}{\mu^2}$$

unde TD=o măsură a erorilor totale ce se pot ivi de-alungul duratei de viaţă a softului (timpul de când incepe proiectarea până când el este scos din uz!), Q_2 = numărul de erori ce rămân după a cea de-a doua etapă de revizuiri şi testări, iar CRE= o măsură a eficienţei cumulate a eliminării erorilor în cele două etape.

Modelul este recomandabil să fie utilizat cu precauţie rezultatele sale putând constitui numai un mod de evaluare preliminar al calităţii produsului soft.

6.4 Introducere formală a metricilor de software

Metricile de complexitate introduse în secţiunile anterioare au o justificare formală [9]. Vom prezenta aici pe scurt argumentele care conduc la definirea metricilor de complexitate.

Măsurătorile se definesc în mod obişnuit ca fiind *atribuirea de valori numerice* unor obiecte astfel încât acestea să reprezinte aceste obiecte în lumina unor *proprietăţi* sau *atribute.* Pentru a introduce măsuri în software trebuie să luăm in considerare trei clase de atribute specifice şi anume: *procesele, produsele* şi *resursele.* Procesele sunt activităţi care au o dimensiune temporară, adică se realizează în timp. O clasă de procese poate fi de exemplu *construirea documentaţiei de specificare* a produsului soft. Produsele sunt obiectele ce se realizează prin procese; de exemplu documentaţiile produse pe parcursul ciclului de realizare a produsului soft. Resursele sunt elementele care *intră* în procese, ca de exemplu personalul de specialitate (indivizi sau echipe), materialele (inclusiv spaţiile pentru birouri şi mobilierul acesora) şi instrumentele de lucru (componente soft sau hard şi metodele utilizate).

Pentru a vorbi de metrici ale căror valori să exprime *calitatea* în exploatare a softului, ar trebui să ne limităm la *produse,* mai precis la *clase de atribute* ale produselor soft, care pot fi *interne* sau *externe* produsului respectiv. Dacă atributele interne sunt uşor de înteles, în cazul unui produs soft, atributele externe depind de anumite entităţi specifice ca: *fiabilitatea* programelor sursă raportată la calculatorul pe care acestea sunt rulate şi la modul particular în care sunt elaborate specificaţiile; gradul de *înţelegere* a documentaţiei de specificare de către utilizatori şi *modul şi posibilitatea de întreţinere* a programului pe parcursul exploatării acestuia (adică *mentenabilitatea*).

Desigur, atributele *interne* se referă la mărimea programului, la funcţionalitatea sa, la structura sa modulară, la corectitudinea logică şi sintactică a programului, etc.

Indiferent ce atribute ale produselor vom lua în considerare cand vrem sa intoducem o metrică de soft, suntem de regulă forţaţi să folosim măsurători ale proceselor care pot aproxima măsuri ale atributelor produselor.

Pe de altă parte trebuie să avem în vedere că atribute externe ca fiabilitatea şi mentenabilitatea se definesc şi determină în funcţie de atribute interne care le influenţează în mod pregnant. In definirea metricilor de software (care să exprime fidel calitatea) trebuie să facem şi o *ordonare* a influenţei atributelor interne asupra atributelor externe. De exemplu, dacă am şti care

din atributele interne (modularitate, complexitatea structurală sau mărimea) au mai mare influenţă asupra mentenabilităţii, atunci am şti care din aceste resurse ar trebui mărite pentru a *prezice* o calitate mai bună în exploatare a produsului.

Din cele precizate până aici rezultă că pentru a introduce *metrici* sau *măsuri* ale softului trebuie să utilizăm modele adecvate pentru fiecare din tipurile de atribute , procese, produse şi resurse utilizate. Va trebui să construim modele formale ne ambigue care să permită construirea unor astfel de metrici.

4.5.1. O teorie a măsurătorilor şi măsuri pentru software.

Vom porni de la o *teorie matematică a reprezentării* măsurătorilor. Pentru aceasta presupunem că se dă o mulţime precizată de entităţi şi un atribut bine precizat. Demersul pentru creiarea unei teorii formale a măsurătorilor trebuie să realizeze următoarele:

- să introducă axiome care corespundă înţelegerii intuitive şi reprezentărilor intuitive despre atribut;

- să fie satisfăcută *o teoremă de reprezentare* care să arate că atributul poate fi reprezentat într-un sistem numeric printr-o transformare care să conserve axiomele;

- să satisfacă *o teoremă de unicitate* care să precizeze că oricare ar fi două funcţii definite pe mulţimea de entităţi cu valori în sistemul de numere ele reprezintă în mod fidel atributul considerat.

Sistemul de numere sau sistemul numeric la care ne-am referit poate fi orice sistem ce rezultă din măsurători (cântăriri, numărări, etc).

Să precizăm mai întâi câteva formalisme privind teoria reprezentării.

Presupunem că se dă o cloasă de obiecte C şi că s-a identificat un atribut Q pe care îl poate avea orice obiect din clasa C. In plus, presupunem că Q induce o mulţime $\mathcal{R}$ de relaţii $R_1, R_2, ..., R_n$ pe C. Cu alte cuvinte, când la nişte obiecte am observat Q aceste obiecte sunt în relatiile R_i.

Să considerăm acum perechea ordonată $\mathcal{L} = (C, \mathcal{R})$.

Pentru a dedfini o măsură a unui atribut avem nevoie de o aplicaţie pe mulţimea de obiecte care posedă atributul cu valori într-un "sistem de numere" de exemplu mulţimea numerelor reale R. Mai general, avem nevoie de *un sistem de relaţii numerice* care constă dintr-o mulţime impreună cu una sau mai multe relaţii pe acea mulţime. Să notăm cu N acea mulţime

$N \subset R$, şi fie

$$\mathcal{P} = (P_1, P_2, ..., P_n) \tag{6.19}$$

o mulţime de relaţii definite pe N. Atunci sistemul de relaţii numerice $\mathcal{R}$ este perechea

$$\mathcal{N} = (N, \mathcal{P}). \tag{6.19'}$$

Pentru a avea o măsură M a unui atribut, avem nevoie să identificăm un sistem de relaţii numerice $\mathcal{N} = (N, \mathcal{P})$ ale cărui relaţii $\mathcal{P}$ "corespund" relaţiilor empirice $\mathcal{R}$ prin aplicaţia $M : C \mapsto N$.

Deci M aplică obiecte din C în elemente din N şi de asemenea aplică relaţii din $\mathcal{R}$ în nrelaţiile coresounzătoare din $\mathcal{P}$. Această aplicaţie M trebuie să asigure că toate relaţiile empirice sunt conservate în sistemul de relaţii numerice. Trebuie deci ca M să reprezinte un *homomorfism*. Pentru a defini formal o măsură considerăm deci C mulţimea obiectelor, fiecare conţinând atributul Q astfel incât

$$\mathcal{R} = (R_1, R_2, ..., R_n) \tag{6.20}$$

este mulţimea relaţiilor din C definite de Q şi

$$\mathcal{P} = (P_1, P_2, ..., P_n) \tag{6.20'}$$

este o mulţime de relaţii în sistemul relaţional numeric $\mathcal{R}$. Spunem că

$$M : (C, \mathcal{R}) \mapsto (N, \mathcal{P}) \tag{6.21}$$

unde $M : \mathcal{L} \mapsto \mathcal{N}$, este o *măsură* pentru Q dacă M este o funcţie de la C la N astfel incât

$$M(R_i) = P_i$$

şi

$$R(x_1, x_2, ..., x_k) \quad daca \quad si \quad numai \quad daca \quad P_i(M(x_1), M(x_2), ... M(x_k)).$$

Ultima condiţie este *condiţia de reprezentare*. Când această condiţie este îndeplinită, atunci atributul în cauză este complet caracterizat.

Condiţia de reprezentare cere ca măsura să stabilească o corespondenţă între obiectele din C (sau mai precis dintre elementele unui model al lui C) şi numere în aşa fel încât relaţiile induse de Q pe C să implice şi să fie implicate de relaţiile dintre imaginile lor în mulţimea de numere.

Având un homomorfism (numit de asemenea *reprezentare*), tripletul $(\mathcal{L}, \mathcal{N}, M)$ este numit *scală*. Când $\mathcal{L}$ şi $\mathcal{N}$ sunt subinţelese din context, ne vom referi la M ca fiind o scală.

Prima problemă importantă în teoria măsurătorii este *problema reprezentării.* Fiind dat un sistem relaţional observat $\mathcal{L}$ şi un sistem numeric relaţional "potrivit" $\mathcal{N}$, să se găsească condiţii necesare şi suficiente pentru existenţa unui homomorfism de la $\mathcal{L}$ la $\mathcal{N}$. Aceste condiţii sunt numite *axiome* de reprezentare şi teorema care stabileşte suficienţa lor se numeşte *teoremă de reprezentare.*

Cea de-a doua problemă a teoriei măsurătorii este problema *unicităţii.* Homomorfismul dat de condiţia de reprezentare nu este în general unic. Dacă de exemplu am vrea să definim o teorie a măsurătorii înălţimii unor oameni şi M ar fi înălţimea, atunci aceasta ar putea fi considerată în centimetri, metri, etc., în funcţie de scala de măsurători aleasă.In fapt, dacă M este o măsură a înălţimii, atunci şi αM este o măsură a înălţimii, oricare ar fi $\alpha, 0 \leq \alpha \leq \infty$. Spunem ^ in acest caz că αM este o *re-scalare* a lui M. Care re-scalare este permisă, acest fapt depinde de proprietăţile sistemului relaţional $(C, \mathcal{R})$ şi acea re-scalare (caracterizată de *teorema de unicitate*) determină ce fel de scală este M. O teoremă de unicitate impune limitări asupra operaţiilor matematice ce se pot efectua cu valorile măsurate, adică cu imaginile lui M în N. De regulă se pot efectua operaţii ca adunarea, calcule cu medii, calculul logaritmului numerelor reale, iar problema principală este dacă rezultatele acestor operaţii dau informaţii *semnificative* asupra obiectelor ce se măsoară.

•**Scale regulate şi semnificaţie.** Teoria tipurilor de scale şi semnificaţia operaţiilor precizează un mecanism cu care putem raţiona şi interpreta anumite aserţiuni asupra sensului măsurătorilor pe care le efectuăm.

Să considerăm de exemplu următoarele aserţiuni referitoare la aprecierea programelor:

1. Numărul de erori detectate la testarea programului X a fost de cel puţin 100.

2. Costul delimitării şi eliminării fiecărei erori din programul X este de cel puţin 100.

3. Detectarea unei erori semantice necesită un timp de două ori mai mare ca detectarea unei erori sintactice,

4. O eroare semantică este de două ori mai complexă decât o eroare sintactică.

Pornind de la un nivel pur intuitiv, aserţiunea 1 pare a avea sens, în timp ce de-a doua nu, deoarece numărul de erori poate fi specificat fără a avea în vedere o scală pe când costul eliminării unei erori presupune obligatoriu o scală de valori. Aserţiunea 3 pare să aibă sens (chiar dacă gândindu-ne bine poate să nu fie adevărată) în timp ce aserţiunea 4 nu are sens deoarece timpul necesar eliminării unei erori este acelaşi indiferent de scala de valori utilizată (adică dacă eliminarea unei erori semantice necesită de două ori mai multe minute decât eliminarea uneia sintactice, ea va necesita tot de două ori mai multe ore, secunde sau zile etc). Deci raportul "complexităţii" (care oricum este ambiguu în aserţiunea 4) nu este în mod necesar acelaş.

Teoria măsurătorii ne permite să determinăm *semnificaţia* unor aserţiuni de tipul celor de mai sus. Această determinare se bazează pe unicitatea homomorfismelor care satisfac condiţia de reprezentare. Am văzut că este posibil să existe mai mult de un homomorfism de la un sistem relaţional empiric $\mathcal{L}$ la un sistem relaţional numeric $\mathcal{N}$. Ceeace dorim să determinăm este modalitatea de a transforma un homomorfism acceptabil într-un alt homomorfism acceptabil. O astfel de informatie ne va spune de exemplu când o aserţiune de tipul 3 de mai sus este semnificativă.

Pentru a simplifica discuţia pe această temă, ne vom restrânge la sistemele relaţionale numerice a căror mulţime de bază este R. In acest caz scala $(\mathcal{L}, \mathcal{N}, M)$ se numeşte *scală numerică.*

Este necesar acum să introducem noţiunea de *transformare admisibilă.* Să considerăm de exemplu măsurarea (în sens obişnuit) a corpurilor fizice. Fiind dată o modalitate de măsurătoare numerică M. adică o aplicaţie ce satisface condiţia de reprezentare pentru toate relaţiile de lungime, este uşor de văzut că orice multiplu scalar αM al lui M satisface de asemenea condiţia de reprezentare (dacă $\alpha > 0$.) Transformarea Φ care aplică pe M în αM este numită admisibilă deoarece prin transformarea măsurătorii numerice, încă se conservă condiţia de reprezentare. Suntem deci în măsură să dăm următoarele definiţii:

Definiţia 6.1. *Considerăm $M : \mathcal{L} \mapsto \mathcal{N}$ este un homomorfism unde $\mathcal{L} = (C, \mathcal{R})$. Presupuunem că Φ este o funcţie care aplică domeniul lui M adică mulţimea*

$$M(C) = \{M(c) : c \in C\} \tag{6.22}$$

în mulţimea N. Atunci compunerea $\Phi o M$ este o funcţie de la C la N. Dacă $\Phi o M$ este un homomorfism de la $\mathcal{L}$ la $\mathcal{N}$, vom numi Φ o transformare ad-

misibilă de scală.

Definiţia 6. 2. *Dacă* $(\mathcal{L}, \mathcal{N}, M)$ *este o scală astfel încât pentru orice altă scală* $(\mathcal{L}, \mathcal{N}, M')$ *există o transformare* $\Phi : M(C) \mapsto N$ *pentru care* $M' = \Phi oM$, *atunci scala* $(\mathcal{L}, \mathcal{N}, M)$ *este numită* regulată. *Dacă orice homomorfism* M *de la* $\mathcal{L}$ *la* $\mathcal{N}$ *este regulat, vom numi* reprezentarea $\mathcal{L} \mapsto \mathcal{N}$ regulată.

Deci, o reprezentare $\mathcal{L} \mapsto \mathcal{N}$ este regulată dacă fiind date două scale M şi M', fiecare poate fi aplicată în cealaltă printr-o transformare admisibilă. Cele mai multe scale întâlnite în metricile de software sunt regulate. Pentru reprezentările regulate avem următoarea definiţie a semnificaţiei.

Definiţia 6. 3. *O aserţiune referitoare la scale numerice este semnificativă dacă este invariantă la toate transformările admisibile.*

Dacă o reprezentare $\mathcal{L} \mapsto \mathcal{N}$ este regulată, adică toate scalele $(\mathcal{L}, \mathcal{N}, M)$ sunt regulate, atunci *clasa transformărilor admisibile* defineşte cum o scală este unică şi astfel de transformări admisibile pot fi utilizate la definirea *tipurilor de scale.* Cele mai importante tipuri de scale sunt rezumate in tabelul de mai jos.

Transformări admisibile	Tipuri de scală	Exemple(interpretări)
$M' = F(M)$ (F este o aplicaţie 1-1)	Nominal	Etichete
$M' = F(M)$ (F-monoton crescătoare) $M(x) \geq M(y) \mapsto M'(x) \geq M'(y)$	Ordinal	Preferinţă, putere,tărie scoruri comparative, teste de inteligenţă.
$M' = \alpha M + \beta, (\alpha > 0)$	Interval	Timp (calendar) temperatură, greutate, (cf. standarde), etc.
$M' = \alpha M, (\alpha > 0$	Raport	Interval de timp, lungime, temperatură (măsurători absolute).
$M' = \alpha M$	Absolută	Numărători

Tabel. Scale de măsurători

Astfel, de exemplu, dacă o familie de transformări este închisă pentru aplicaţii $1-1$, atunci tipul de saclă este *nomonal;* dacă acea famile este inchisă faţă de înmulţirea cu un scalar (cum este de ex. *înălţimea*) atunci scala este de tipul *raport.* Acum putem decide care operaţii pot fi realizate in mod semnificativ asupra unor măsurători date.

Aspectele matematice ale teoriei măsurătorilor sunt strâns legate de teoreme care stabilesc condiţiile în care anume scale de măsurători directe sunt posibile pentru anumite sisteme relaţionale numerice. Un exemplu tipic este acela al teoremei următoare, datorată lui Cantor, care dă condiţiile necesare şi suficiente pemtru măsurăstori ordinale într-o clasă importantă de sisteme relaţionele pe o mulţime numărabilă.

Teorema 6.1. *Presupunem că C este o mulţime numărabilă şi R este o relaţie binară pe C. Atunci există o funcţie M cu valori reale definită pe C care satisface condiţia*

$$xRy \mapsto M(x) > M(y) \tag{6.23}$$

dacă şi numai dacă (C, R) este o ordine slabă strictă. In plus, dacă există un astfel de M, atunci $\mathcal{L} = (C, R) \mapsto \mathcal{N} = (\mathbf{R}, >)$ este o reprezentare regulată şi $(\mathcal{L}, \mathcal{N}, M)$ este o scală ordinală.

Observaţia 6.1. *O ordine slabă strictă R pe C* este *asimetrică ($xRy \mapsto \neg(yRx) \forall x, y \in C$) şi* negativ- tranzitivă *($xRy \mapsto xRz$ sau $zRy, \forall x, y, z \in C$).*

Acest rezultat abstract are importante ramificaţii în studiul *măsurilor de complexitate a programelor.* Au fost făcute multe încercări de a se găsi o singură funcţie cu valori reale care să caracterizeze complexitatea programelor. Una din măsurile de complexitate utilizate este aşa cum am văzut *numărul ciclomatic al lui McCabe.* Programele formează o clasă de obiecte C şi astfel funcţiile decomplexitate definite sunt măsuri (ordinale) ale "complexităţii" dacă şi numai dacă relaţia R: "x este mai complex decât y" este o ordine slabă strictă. Se pare că nu există o noţiune de complexitate generală cu această proprietate deoarece tranzitivitatea negativă nu are de regulă loc. De ex. în *Fig.6.1* pare plauzibil că xRy dar nici xRz nici xRy nu au loc.

Totuşi dacă am presupune că complexitatea ciclomatoică M ar fi o măsură de complexitate valabilă, atunci deoarece $M(x) = 3$, şi $M(y) = 2$, ar trebui ca xRz, adică x este mai "complex" decât z. Dar nu este clar că x este mai "complex"; deci din acest punct de vedere nu putem afirma că M este o "măsură de complexitate".

Teorema precedentă care dă o legitimitate a măsurilor de complexitate, ea poate fi la fel utilizată pentru a construi scale ordinale de măsură pentru complexitatea atributelor (ca de exemplu căile de control a fluxului în programe structurate) care conduc în mod rezonabil la o ordine slabă strictă. Este însă necesar

să stabilim mai întâi o ordine a documentelor, deoarece atunci măsura ordinală rezultă în mod natural.

Una din cele mai puternice aplicaţii ale teoriei tipurilor de scală şi semnificaţiei este de a determina care tipuri de operaţii sau analize statistice pot fi aplicate unor tipuri particulare de măsuri.

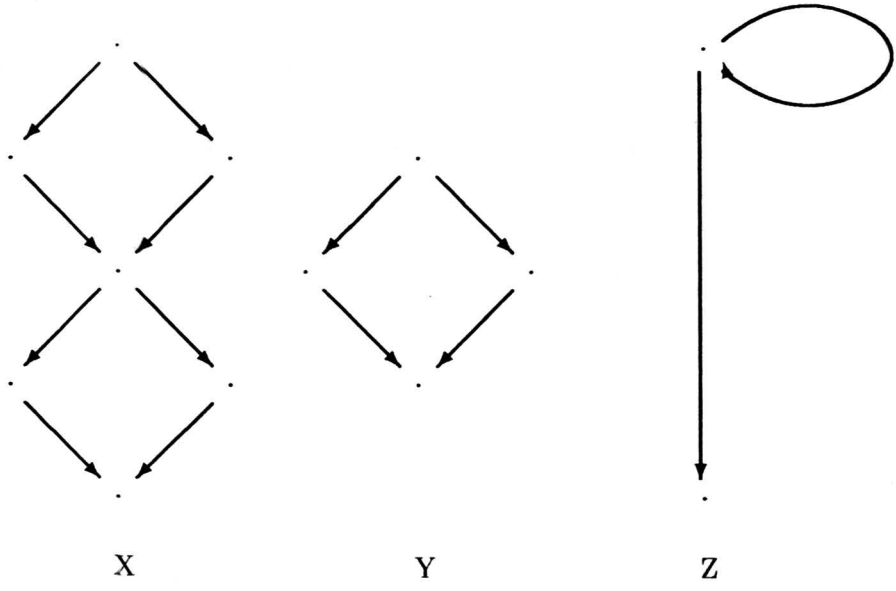


*Fig.*6.1.

Ne vom opri la cea mai simplă prelucrare statistică ce se poate face cu nişte măsurători date şi anume calculul unei *medii*. Presupunem că M este o măsură şi că $X = (x_1, x_2, \ldots, x_n)$ şi $Y = (y_1, y_2, \ldots, y_m)$ sunt două seturi de măsurători pentru care cunoaştem $M(x_i)$ şi $M(y_j)$. Vrem să determinăm dacă măsura medie a lui X este mai mare decât măsura medie a lui Y. Mai precis vrem să vedem dacă aserţiunea

"media lui $M(x_i)$ este mai mare decât media lui $M(y_j)$"

este semnificativă, sau cu alte cuvinte (considerând ca medie pe cea *aritmetică*) vrem ca aserţiunea

$$\frac{1}{n}\sum_{i=1}^{n} x_i > \frac{1}{m}\sum_{j=1}^{m} y_j \tag{6.24}$$

este semnificativă. Pentru a fi semnificativă ar trebui ca pentru orice transformare

admisibilă Φ, relaţia (1) să aibă loc dacă şi numai dacă

$$\frac{1}{n}\sum_{i=1}^{n}(\phi oM)x_i > \frac{1}{n}\sum_{j=1}^{m}(\Phi oM)y_j. \tag{6.25}$$

Este uşor de văzut că dacă Φ este o transformare de tipul $\Phi(x) = \alpha x, \alpha > 0$ sau chiar de tipul $\Phi(x) = \alpha x + \beta, \alpha > 0$, atunci relatia (1) este adevărată dacă şi numai dacă relaţia (2) este adevărată. Deci relaţia (1) este semnificativă dacă M este fie o scală de tip raport fie o scală de tip interval. Pentru a constata acest lucru să presupunem că avem o scală de complexitate ordinală a softului care cuprinde clasele următoare de complexitate

Triviala	1	1
Simplă	2	2
Moderată	3	3
Complexă	4	4
Foarte complexă	5	10

Ultimele coloane reprezintă valorile a două scale M_1 şi M_2 respectiv. Exsistă o transformare monotonă a lui M_1 în M_2, anume $\{1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 3, 4 \mapsto 4, 5 \mapsto 10\}$. Acum să presupunem că avem produsele software S_1, S_2, S_1', S_2' pentru care $M_1S_1) = 3, M_1(S_2) = 4, M_2(S_1') = 1, M_2(S_2') = 5$. Am spera să putem spune care din cele două mulţimi de programe $\{S_1, S_2\}$ $\{S_1', S_2'\}$ are cea mai mare complexitate medie: totuşi aceasta nu este semnificativă deoarece

$$3.5 = \frac{1}{2}(M_1(s_1) + M_1(S_2)) > \frac{1}{2}(M_1(S_1') + M_1(S_2')) = 3$$

în timp ce

$$3.5 = \frac{1}{2}(M_2(S_1) + M_2(S_2)) < \frac{1}{2}(M_2(S_!') + M_2(S_2')) = 5.5.$$

Deoarece ultima relaţie derivă din prima primtr-o transformare admisibilă a lui M_1, cea de-a doua nu este amisibilă căci inegalitatea nu se menţine în cea de-a doua relaţie.

Dar nu este totul pierdut din cauză că există o măsură de "medie" care este semnificativă la o scală ordinală de măsuri. Aceasta este valoarea *mediană* adică valoarea de la mijloc din sirul valorilor ordonate.

Să mai observăm că pentru o scală nominală de măsură, mediana nu este nici ea o măsură medie semnificativă, dar *modul* (cea mai frecventă valoare din şirul de date) este semnificativă.

4.5.2 Abordarea structurilor interne ale programelor.

Aşa cum am văzut, metodele de abordare în ingineria programelor au la bază reguli, tehnici şi instrumente de producere a softului care se bazează pe *structura ierarhică a programelor.* Trebuie să se ţină seama atât de structura *internă* cât şi de cea *externă.*

Este important să vedem cum se pot defini măsuri ale atributelor interne ale programelor. Ne vom referi la măsura unui atribut important de proiectare şi anume **cuplarea.**

Să considerăm atribute ale produsului soft pentru care modulele există sau sunt aparente.Exemple de acest tip sunt programele sursă şi poate mai importante documentaţiile de proiectare în care se descriu modulele sistemului şi interrelaţiile dintre ele. Deoarece un document de proiectare sau de specificare ce descrie un-sistem software este disponibil inainte de implmentare (aşa cum prevăd aproape toate metodologiile moderne de proiectare!), orice măsură a atributelor importante ale acestor documente poate furniza informaţii utile despre sistem chiar inainte de dezvoltarea acestuia. Vom presupune că produsele pe care le analizăm sunt documente de proiectare care detaliază interrelaţiile dintre module.

Cuplarea poate fi definită ca o măsură a gradului de interdependenţă dintre module. Definiţia este ambiguă deoarece este neclar dacă cuplarea este un atribut global de proiectare or un atribut al fiecărei perechi de module. Vom studia cuplarea între perechi de module, întrucât *cuplarea globală* trebuie să derive din prima. Definiţia cuplării dată mai sus mai are un neajuns: deşi este numită "măsură" nu precizează o caracterizare numerică a atributului de cuplare. Acest lucru pare ciudat deoarece empiric se pare că există relaţii bine stabilite care ar indica faptul că există cel puţin o scală ordinală de măsurători. Vom preciza mai întâi câteva relaţii binare care există pentru o pereche de module x, y şi anume:

R_5 : $(x, y) \in R_5$ dacă x se referă la *interiorul* lui y, adică dacă el (x) intră în, schimbă date sau altrerează o instrucţiune din y. Să numim tipul de cuplare caracterizat de R_5 *cularea conţinutului.*

R_4 : $(x, y) \in R_4$ dacă x şi y se referă la aceleaşi date globale. Acest tip de cuplare R_4 se va numi *cuplare comună.* Acest tip de cuplare nu este bun deoarece dacă formatul datelor globale trebuie modificat vor trebui modificate toate modulele în relaţie de cuplare comună.

R_3 : $(x, y) \in R_3$ dacă x transferă un parametru lui y cu intenţia de a-i controla

comportarea, adică parametrul este un semnal. Acest tip de cuplare se numeşte *cuplare controlată.*

R_2 : $(x, y) \in R_2$ dacă x şi y acceptă acelaş tip de înregistrare ca parametru. Acest tip de cuplare R_2 s-ar putea numi cuplare "ştampilată" sau mai bine *etichetată.* O astfel de cuplare introduce o interdependenţă intre module care altfel nu ar avea nicio legătură.

R_1 : $(x, y) \in R_1$ dacă x şi y comunică prin parametri, fiecare fiind ori elemente de date singulare ori seturi de date omogene care nu incorporează niciun element de control. Tipul de cupalre R_1 va fi numit *cuplare prin date* şi el este necesar pentru orice comunicare între module.

R_0 : $(x, y) \in R_0$ dacă x şi y nu comunică între ele, adică sunt total independente. Acest tip de cuplare va fi numit *ne-cuplare.*

Cu această clasificare a tipurilor de cuplare, putem imagina o ordine empirică a acestor tipuri ca în Fig.6.2 unde ordinea crescătoare a cuplării merge de sus în jos.

In acest moment avem nevoie de un model precis de studiere a cuplării. Modelul pe care îl alegem este acela al unui *multigraf orientat şi etichetat* adică al unui graf orientat şi etichetat care poate avea mai multe arce (orientate!) între două noduri. Eticheta fiecărui arc este o pereche ordonată în care prima componentă reprezintă tipul de cuplare (menţionat în Fig.6.2) şi cea de-a doua componentă reprezintă numărul de cuplări de acel tip care apar intre noduri (in sens direct!).

5.	Cuplarea conţinutului	rău
4.	Cuplarea comună	↑
3.	Cuplarea controlată	↑
2.	Cuplarea etichetată	↓
1.	Cuplare prin date	bine
0.	Ne-cuplare	

Fig.6.2.Tipuri de cuplare

In Fig. 6. 3 avem o parte a unui model de graf de cuplare. Acest graf reprezintă patru module unde:

- modulele M_1 şi M_2 folosesc două tipuri diferite de înregistrări;
- modulul M_1 transmite modulului M_3 un parametru care acţionează ca o etichetă in M_3;

- modulul M_2 trimite o ramificare spre M_4 şi de asemenea transmite doi parametri ce acţionează ca o etichetă in M_4.

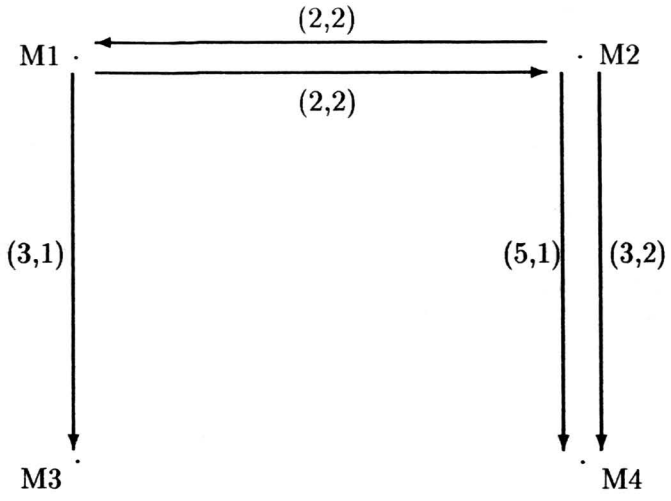


Fig.6.3. Graf de cuplare.Exemplu.

Considerăm relaţia " < " definită pentru perechi de module; pentru perechile de module $(x, y), (x', y')$ relaţia este

$$(x, y) < (x', y') \tag{6.26}$$

când există un nivel de cuplare mai tare pentru (x', y') decât între (x, y). Această relaţie are proprietatea că dacă (x_i, y_i) este o pereche de module in R_i pentru $i = 0, 1, ..., 5$, atunci

$$(x_0, y_0) < (x_1, y_1) < (x_2, y_2) < (x_3, y_3) < (x_4, y_4) < (x_5, y_5)$$

adică orice pereche de module cuplate după conţinut are un nivel de cuplare decât orice pereche cuplată în comun, etc.

Cea mai simplă măsură a unei perechi de cuplări care conservă relaţiile menţionate va fi o transformare de forma

$$M(x, y) = i, \quad 0 \leq i \leq 5 \tag{6.27}$$

adică măsura cuplării pentru module care sunt ne-cuplate este 0, pentru module cuplate prin date este 1, etc,..., pentru cele cuplate după conţinut este 5.

O problemă ce apare frecvent într-o asemenea abordare este că noi nu putem să ţinem cont de diferitele tipuri de cuplări care se pot ivi din cele şase clase identificate. De exemplu intuitiv se pare că dacă (x,y) şi (x',y') sunt ambele perechi de module cuplate după conţinut, adică sunt în R_5 şi dacă x se referă o singură dată la ceva interior lui y iar x' se referă de mai multe ori la ceva din y', atunci nivelul de cuplare dintre (x',y') este mai puternic decât cel dintre (x,y), adică

$$(x,y) < (x',y').$$

Pentru a implementa o astfel de măsură avem nevoie să numărăm interconexiunile de fiecare tip dintre perechi de module. Aceasta ne va permite să stabilim măsuri pentru fiecare tip de cuplare, de exemplu volumul de cuplări după conţinut ar putea fi şapte interconexiuni; dar acest mod de a număra nu sugerează uşor că ar putea fi vorba de o singură măsură de cuplare între module.

Dacă dorim să avem o singură măsură a cuplării dintre modulele x, y care să conserve toate relaţiile identificate până acum, atunci aceasta ar fi următoarea măsură de scală ordinală:

$$M(x,y) = i + \frac{n}{n+1} \tag{6.28}$$

unde i este măsura de cuplare cea mai mare de tipul de mai sus (adică $i = 3$ dacă x şi y sunt cuplate prin control şi n este numărul de interconexiuni dintre x şi y).

După ce am stabilit câteva măsuri rezonabile din punct de vedere intuitiv, care măsoară cuplarea între perechi de module, să ne indreptăm atenţia către studiul *cuplării globale* care poate fi privită şi ca noţiunea de *conectivitate* a diagramei sau schemei de structură a unui program. Din nou vom porni de la relaţiile empirice existente pentru acest atribut. O Astfel de relaţie poate fi exprimată intuitiv prin următoarea axiomă a cuplării:

Axiona 1. Dacă D şi D' sunt două diagrame modulare de structură astfel încât diferenţa dintre ele constă în faptul că D' conţine în plus faţă de D un arc de conexiune, atunci D' are o cuplare mai mare decât D.

Dacă dorim să definim o mšură care să conserve această axiomă, atunci ar trebui să însumăm pentru toate perechile de module din diagrama de structură măsurile cuplării definite mai sus. O atare definiţie s-ar reduce la ceva separat care exprimă mai degrabă "volumul" programului. Se pare că cuplarea globală este mai bine caracterizată ca o "medie" a cuplării perechilor, care ar putea fi exprimată de următoarea axiomă.

Axioma 2. Presupunem că S este un sistem care constă din modulele D_1 şi D_2. Dacă S este extins la S' prin adăugarea unui modul D_3 şi dacă cuplarea lui

D_1, D_3 este egală cu cuplarea inițială a lui D_1, D_2 atunci cuplarea globală a lui S este egală cu cea a lui S'.

Să observăm că "media" creşte dacă se adaugă exact un arc (ca in axioma 1) dar nu creşte în mod necesar dacă dacă un alt modul cu noi interconexiuni este adăugat. In fapt cuplarea glogală poate chiar să scadă in acest caz.

In acest exemplu controlul cuplării între două module este "redus" la cuplarea de date prin introducerea unui nou modul cu două noi arce. Există două modalităţi de a judeca cum se reduce cuplarea:

1. deoarece controlul total al cuplării este acum zero şi controlul cuplării "domină" cuplarea datelor.
2. deoarece cuplarea "medie" între module sa reduce.

Cea de-a doua modalitate pare a fi mai adecvată dar şi aici este o problemă. Am văzut că nu putem ajunge mai bine decât la o scală ordinală pentru noţiunea generală de cuplare şi am observat mai sus că nu există o noţiune semnificativă de medie *in sensul mediei aritmetice* pentru o asemenea măsură. Totuşi există o noţiune semnificativă de medie dacă o luăm drept *mediana* valorilor.

In acest fel putem defini intuitiv o măsură rezonabilă globală de cuplare, ca nivel mediu pentru sistem adică:

Cuplarea globală a unui sistem S constând din modulele $\{D_1, D_2, ..., D_n\}$ este dată de valoarea mediană a mulţimii $\{M(D_i, D_j) : 1 \leq i < j \leq n\}$ unde M este măsura cuplării unei perechi definită mai sus.

Problema cuplării este de considerat deoarece s-a constatat practic că programele care conţin multe erori în *interfeţele* lor, au un număre de erori ce depinde de mărimea cuplărilor. Ideile tradiţionale privind măsurarea cuplării se bazează pe următoarele măsuri care se referă mai degrabă la modulele individuale decât la perechi:

- numărul maxim de interconexiuni per modul;
- numărul mediu de interconexiuni per modul;
- Numărul total de interconexiuni per modul;
- numărul de module care accesează interconexiuni de control;
- numărul de structuri de date interconectate la nivelul modulului de cel mai inalt nivel (modulul principal!).

Pe baza acestor măsuri de cuplare s-a formulat ipoteza:

Programele cu valoarea cuplării mare conţin mai multe erori dacât cele cu valoarea cuplării mică.

Ca ipoteza să fie satisfăcută ar trebui ca măsurile de cuplare bazate pe numărători enumerete anterior să se coreleze puternic (din punct de vedere statistic) cu numărul de erori. Ori acest lucru nu se intâmplă.

Folosind măsura cuplării bazată pe perechi de module în [10] se sugerează o ipoteză interesantă, mult mai realistă.

Programele cu aceeaşi lungime şi aceleaşi nivele de funcţionalitate dar cu valori ale măsurii de cuplare mai mari conţin mai multe erori.

References

[1] ALIREZA AZEM. (1995). *Software Reliability Determination for Conventional and Logic Programming.* Walter de Gruyter, Berlin, N.Y.

[2] ASHER, HAROLD and FEINGOLD, HARRY. (1984). *Repairable System Reliability,* Marcel Dekker Inc., New York, Basel.

[3] BARLOW,R., PROCHAN, F. (1965). *Mathematical Theory of Reliability,* John Wiley, New York.

[4] BARLOW,R.,PROCHAN,F. (1975). *Statistical Theory of Reliability and Life Testing,* Holt, Reinhart and Winston, New York.

[5] BASILI, V. R., PERRICONE, B.T. (1984). "Software Errors and Complexity: an Empirical Investigation", *Comm. ACM, Vol. 27, No.1, p.42-51.*

[6] BIROLINI, A. (1994). *Quality and Reliability of Technical Systems. Theory-Practice-Management,* Springer Verlag.

[7] CATUNEANU,V.M., MOLDOVAN,C., POPENŢIU,F., POPOVICI, D. (1991). "Software Reliability Release Policy with Testing Effort", *Microelectron. Reliab., No.5, p. 895-899*

[8] DROESBEKE, J.J, FICHET B., TASSI Ph. (Eds) (1989). *Analyse Statistique des Durées de la Vie.(Modelisation des Données Censurées).* Economica, Paris.

[9] EJIOGU, G. Lem.(1990). "Beyound Struxtured Peogramming:An Introduction to the Principles of Software Metrics", *Structured Programming No. 11, p.27-43, Springer Verlag, New York, Berlin.*

[10] FENTON Norman and MELTON Austin (1990). "Deriving Structurally Based Software Measures". *J. of Systems and Software, 12, p. 177-187.*

[11] FENTON Norman (1992). "When a software measure is not a measure". *Software Engineering Journal, September 1992, p. 357-362.*

[12] GERSTBAKH, I., B. (1989). *Statistical Reliability Theory,* Marcel Dekker Inc., New York and Basel.

[13] HARARY, F. (1972). *Graph.Theory,* Addison-Wesley Pub.Co. Inc., Reading Massachuseths - Menlo Park.

[14] HALSTEAD, M. H. (1977). *Elements of Software Science.* Elsevier, New York.

[15] HOYLAND ARNJOLT and RAUSAND MARVIN. (1994). *System Reliability Theory. Models and Statistical Methods,* John Wiley and Sons, New York.

[16] JELINSKI, Z. and MORANDA, P.B. (1972). "Software reliability research", in *Statistical Computer Performance Evaluation, Ed.W.Freiberger, Academic Press, New York, p.465-497.*

[17] KAI-YUAN CAI. (1997). " Censored Software Reliability Models". *IEEE Transaction on Reliability, Vol.46, No.1.*

[18] KITCHENHAM BARBARA. (1990)."Software Development Metrics and Models, Appendix C", *Software Reliability Handbook,PAUL ROOK Ed., Elsevier Applied Sc., London, New York, p.441-486.*

[19] KITCHENHAM BARBARA. (1990)."Software Development Cost Models, Appendix D", *Software Reliability Handbook, PAUL ROOK Ed.,Elsevier Applied Sc.. London. New York. p. 487-517.*

[20] LANGBERG, N. and SINGPURWALLA, N. D. (1985). "A Unification of some Software Reliability Models", *SIAM J. Scientific and Statistical Computation, 6. p.781-790.*

[21] LINDLEY, D.V. and SINGPURWALLA, N. D. (1986). "Multivariate distributions for the life lengths and components of a system sharring a common environment", *J.Appl.Probab., 23, 418-431.*

[22] LITTLEWOOD, B. and VERAL, J. L. ((1973). "A Reliability Growth Model for Computer Software", *Applied Statistics, 22, p.332-346.*

[23] LIXĂNDROIU, DORIN. (2001). *Fiabilitatea sistemelor. Modele şi algoritmi.* Ed.Dacia, Cluj-Napoca.

Tiparul s-a executat sub c-da nr. 1002/2002 la
Tipografia Editurii Universităţii din Bucureşti

[24] LYN KUO, JAE CHANG LEE, KIHEON CHOI and TAE YOUNG YANG. (1997). "Bayes Inference for S-shaped Software Reliability Models", *IEEE Transactions on Reliability, Vol 46, No. 1, p. 76-80.*

[25] MUSA,J., IANNINO, A. and OKUMOTO, K. (1987). *Software Reliability,* McGrow-Hill, New York.

[26] SHAKED MOSHE, SHANTIKUNAR GEORGE J. (1990). "Reliability and Maintenability". *Handbook in OR & MS, Vol 2, D.P. Heyman, N.J. Sobel Eds.,Elsevier Science Publishers B .V. (North Holland).*

[27] SHERWIN, D.J. and BOSCHE, A. (1993). *The Reliability, Availability and Productivness of Systems.* Chapman and Hall, London.

[28] SLUD ERIC (1997). "Testing for imperfect debugging in software reliability". *Scand. J. Statist., 24, p. 555-572.*

[29] ŞTEFĂNESCU, ŞTEFAN and VĂDUVA, Ion. (1984). "Asupra unor modele de analiza fiabilităţii programelor", *Lucr. Coloc. INFO-Iaşi'84, p. 152-162.*

[30] ŞTEFĂNESCU, ŞTEFAN and VĂDUVA, ION. (1985). "Modele pentru analiza fiabilităţii programelor", *Lucr. Coloc. INFO-Iaşi'85, p.135-151.*

[31] TAPAN KUNAR NAYAK. (1987). "Multivariate Lomax distribution: properties and usefulness in reliability theory".*J.Appl.Prob., 24, 170-177.*

[32] VĂDUVA, I. (1999). "Simulation of systems reliability", *Proc. IEPM'99, International Conference on Industrial Engineering and Production Management, Glasgow, July 12-15, Book1. p.201-210.*

[33] VĂDUVA, I. (1999). "On Bayessian models for software reliability", Proc. Annual Conf. Math.Soc. of Romania, Craiova, 27-30 May (sub tipar).

[34] VĂDUVA, I. (1977). *Modele de simulare cu calculatorul,* Editura Tehnică, Bucureşti.

[35] XIE, M. (1991). *Software Reliability Modelling.* World Scientific, Singapore, London.

[36] YAMADA, SHIGERU; OHTERA, HIROSHI and HARISHA, HIROYUKI. (1986). "Software Reliability Growth Models with Testing Effort", *IEEE Transactions on Reliability, Vol.R-35, No.1, p. 19-23.*