Costurile fiabilitatii software

Mihai POPESCU, Viorel IONESCU
Academia Tehnica Militara

Între costul realizarii unui produs software si nivelul fiabilitatii alocat în etapele de elaborare, exista o relatie foarte strânsa. Marimea costurilor implicate e direct proportionala cu cerinta impusa fiabilitatii. La costurile având ca origine existenta erorilor (de specificatie, design, codificare) se adauga si cheltuielile de exploatare si întretinere. Doua tendinte au stimulat puternic cererea pentru masurarea fiabilitatii software: o crestere continua a sistemelor de operare în timp real, impactul costurilor fiind extrem de puternic în cazul nefunctionalitatii (exemplu: defectarea unui sistem de rezervare aerian); cresterea prelucrarilor distribuite si în retea au marit foarte mult dimensiunea si complexitatea sistemelor de calcul, multe fiind multiprocesor, cu un numar mare de componente software diferite ce lucreaza simultan si interactiv, putând genera probleme legate de planificarea procesorului, a proceselor si de ocupare concurentiala a altor resurse (memorie, periferice).
Cuvinte cheie: *productivitate, scala de apreciere, drivere de cost.*

1. Modelul general al costurilor si drivere de cost

1.1. Modele de cost

Majoritatea modelelor de cost presupun o relatie de baza între efort si dimensiune caracterizata printr-o constanta multiplicativa *a* si un termen exponential *b* de forma [1]:
$Effort = a*(dimensiune)^b*d_1*d_2*...*d_n$. Termenul *a* este cunoscut în literatura de specialitate sub denumirea de "productivitate unitate" si furnizeaza scala de conversie între dimensiune si efort, iar *b* identifica nivelul de acoperire introdus de dimensiunea produsului; *b* este de obicei apropiat de 1. O valoare a lui *b* mai mare decât 1 semnifica faptul ca se cere un efort relativ mai mare pentru a realiza proiecte de anvergura (scala neeconomica). Pentru *b* mai mic decât 1 se spune ca exista o economie de scala. Majoritatea modelelor publicate (inclusiv COCOMO [4]) au folosit b>1, presupunând ca productia software cere scala neeconomica.
Ecuatia de baza este ajustata cu alti factori numiti *drivere de cost* (d_i, i=1,2,...,n), care se folosesc pentru a pondera relatii dimensiune-efort conform unor conditii specifice unui proiect particular. Acestia sunt factori ca: complexitate produs; capabilitate conducere; fiabilitate ceruta s.a. Ei sunt masurati folosind o scala de metrici ordinala, cu puncte ca: "foarte scazut", "scazut", "mediu", "înalt" si "foarte înalt", fiecare punct de scala al fiecarui driver de cost având o valoare numerica asociata, folosita pentru stabilirea nivelului de ajustare.
Driverele de cost sunt factori care se presupune ca influenteaza productivitatea software, aceasta fiind definita prin raportul dintre dimensiunea produsului si efortul personalului care realizeaza proiectul:
Productivitate = Dimensiune / Efort. Pentru a aprecia daca un driver de cost ar trebui sau nu sa fie inclus ca o variabila explicativa într-un model de productivitate, se utilizeaza tehnica numita *analiza variatiei* [2]. Aceasta impune calcularea valorii medii a unui grup de proiecte pentru fiecare valoare pe care driverul poate s-o ia si sa se aprecieze daca sunt diferente semnificative între valorile medii. Referitor la aceasta afirmatie, este relevanta analiza facuta pe proiectul ESPRIT MERMAID, pe doua seturi de date despre driverele de cost potentiale (DS1 si DS2) [4]. DS1 a folosit numararea punctelor functie bazat pe suma numarului de tranzactii si entitati, DS2 a folosit punctele functie Al-brecht conventionale. DS1 a inclus informatii despre 81 proiecte, cu date despre 3 drivere de cost potentiale: expe-

rienta echipei; experienta managerului proiectului; tip mediu de programare (cu trei puncte de scala a metricilor: COBOL conventional, COBOL extins, 4GL). DS2 a cuprins 28 proiecte si 21 drivere de cost, dintre care enumeram: cerinta utilizatorului, experienta utilizatorului cu aplicatia, stabilitatea personalului, timpul de raspuns al sistemului, experienta echipei de dezvoltare, stabilitatea cerintelor, complexitatea problemei, folosirea instrumentelor software, experienta echipei în utilizarea instrumentelor software, familiaritatea cu limbajul de programare, experienta conducatorului de proiect în managementul de proiect, mediul de lucru. În ambele seturile de date, a fost un singur efect major asupra productivitatii, produs de folosirea sau nefolosirea limbajelor de generatia a 4-a. Astfel, contrar a ceea ce se crede în mod obisnuit: experienta personalului nu afecteaza productivitatea; micile îmbunatatiri în folosirea metodelor de dezvolatre nu cresc productivitatea.
COCOMO este unul dintre modelele în care multi vad diferentele în ceea ce priveste personalul de dezvoltare proiect ca un factor de productivitate major. Totusi, analiza setului de date COCOMO nu sustine propunerile construite în model [4].
Dintre factorii de personal încorporati în COCOMO (abilitatea analistului, experienta în dezvoltarea aplicatiei, priceperea programatorilor, experienta cu limbajul de programare, experienta masina virtuala), numai doi (experienta cu limbajul de programare si experienta masina virtuala) au un efect statistic semnificativ în productivitate.
Productivitatea obtinuta în proiecte de personal cu experienta la scala de apreciere "nominala" nu este diferita esential de cea razultata prin experienta "înalta".

1.2.Independenta driverelor de cost

Modelele de cost au un numar mare de drivere de cost care se folosesc pentru a face ajustari ale predictiei modelului de baza. Modelele presupun ca driverele sunt independente asa încât fiecare ajustare poate fi facuta independent de altele. Daca aceasta presupunere nu e adevarata, modelul va fi incomplet si nu va da predictii exacte. Problema afecteaza, de asemenea, modelele "dimensiune", cum ar fi punctele functie, care include un numar mare de factori de ajustare. Pentru a vedea daca driverele de cost sunt independente, nu este necesar sa se testeze fiecare posibila prima interactiune, ci se poate, mai degraba, utiliza o tehnica statistica numita *analiza componentei principale*. Ea transforma un set de *n* variabile într-un nou set, în care fiecare variabila este independenta. Fiecare noua varibila (nd_i, i=1,...,m) este o combinatie liniara a vechilor variabile (di, i=1,...,m):

$$nd_1 = a_{11}*d_1+a_{12}*d_2+...+a_{1m}*d_m$$
$$nd_2 = a_{21}*d_1+a_{22}*d_2+...+a_{2m}*d_m$$
$$...$$
$$nd_m = a_{m1}*d_1+a_{m2}*d_2+...+a_{mm}*d_m$$

Matricea coeficientilor a_{i1}, ... , a_{im}, i =1, ...,m se numeste *de corelatie*, iar vectorul (a_{i1}, ..., a_{im}) se numeste *vectorul Eigen*.

2. Politici de livrare software optimale din punct de vedere al costului

Una dintre problemele de baza care se pun pentru managerii de proiecte software este sa decida timpul potrivit pentru livrarea softului catre utilizator. Aceasta hotarâre este numita o problema de eliberare optimala software, în sensul ca se ia în contextul unor criterii care trebuie îndeplinite. Considerând criterii de evaluare cum ar fi timpul de livrare, costul si fiabilitatea software obtinuta, trebuie sa se estimeze timpul de desfasurare a testarii.
În [3], folosind *modelul de depanare imperfect*, se investigheaza aspecte legate de livrarea optimala a software-ului bazata pe criterii de fiabilitate si cost software.
În cele ce urmeaza, se prezinta politicile considerate esentiale pentru problematica vizând costurile si raportul fiabilitate-cost.
Se definesc urmatorii parametri de cost: C1: costul de depanare pe defect în timpul fazei de testare; C2: costul de depanare pe defect

în timpul fazei operationale $(C_2 \rangle C_1 \rangle 0)$; C3: costul de testare pe unitatea de timp $(C_3 \rangle 0)$.
Presupunem ca numarul asteptat de defecte detectate (eventual) este $M = [N/p]$, unde N este continutul de defecte initial, anterior testarii (necunoscut), p este probabilitatea depanarii perfecte ($0 \le p \le 1$), q=1-p.
Atunci, costul software asteptat total este:

$$C(m) = C_1 m + C_2 (M - m) + C_3 \left[1 - (p/k + q)^m\right] / \left[p \cdot D \cdot (1 - 1/k)\right],\ 0 \le m \le M; \qquad (I)$$

unde D si k sunt rata de defectare initiala si rata descresterii defectelor(D>0, 0<k<1).
În felul acesta, întregul *m* care minimizeaza C(m) este numarul optim de defecte detectate, *m**. Pentru a gasi m=m*, minimizând C(m) în ecuatia de mai sus, se obtine urmatoarea ecuatie:

$$\begin{aligned} y(m) &= C(m+1) - C(m) = \\ &= C_1 m + C_1 + C_2 (M - m - 1) + C_3 \left[1 - (p/k + q)^m (p/k + q)\right] / pD(1 - 1/k) - \\ &- C_1 m - C_2 (M - m) - C_3 \left[1 - (p/k + q)^m (p/k + q)\right] / pD(1 - 1/k) = \\ &= (C_1 - C_2) + C_3 / D(p/k + q)^m (1 - q - p/k) / p(1 - 1/k) \\ &= (C_1 - C_2) + C_3 / D(p/k + q)^m; \quad (0 \le m \le M - 1) \end{aligned} \qquad (II)$$

De observat ca $\psi(m)$ este o functie monoton crescatoare în raport cu m (numarul de defecte detectate); p+q=1 si $0 \langle k \langle 1$.
Daca $?(0) \langle 0$, atunci minimum *m* care sustine $\psi(m) \ge 0$ satisface inegalitatile C(m+1) ≥C(m) si C(m)<C(m-1). Corespunzator, numarul optim de defecte detectate, m*=m, este dat de:

$$m_1 = \ln\{D(C_2 - C_1)/C_3\} / \ln(p/k + q) \qquad (III)$$

Demonstratie:
?(m) =C(m+1)-C(m) este strict crescatore si continua.
cum $\psi(0) < 0$ si $\lim_{m \to \infty} y(m) = \infty$, rezulta ca ∃ m1 a.î. $y(m1) = 0$ si ∀ m > m1, $\psi(m) > y(m1) = 0$.
Dar $y(m1) = 0 \Leftrightarrow C2\text{-}C1 = C3/D(p/k+q)^{m1} \Rightarrow$ m1=ln{D(C2-C1)/C3}/ln(p/k+q).
Astfel, exista urmatoarea teorema pentru o problema de eliberare software optimala pentru cost.
Teorema 1. *Presupunem ca* $C_2 \rangle C_1 \rangle 0$ *si* $C_3 \rangle 0$

(1) Daca $D \rangle \dfrac{C_3}{C_2 - C_1} \ge \dfrac{D}{(p/k + q)^{M-1}}$,

(2) atunci numarul optimal de defecte detectate este $m^* = m_1$ $(1 \le m_1 \le M - 1)$ *si timpul de livrare software optim este:*
$T^* = \left[1 - (p/k + q)^{m_1}\right] / (pD(1 - 1/k))$.

(3) Daca, $\dfrac{D}{(p/k + q)^{M-1}} \rangle \dfrac{C_3}{C_2 - C_1}$, *atunci numarul optim de defecte detectate este m*=M si timpul de livrare software optim este* $T^* = \left[1 - (p/k + q)^M\right] / pD(1 - 1/k)$.

(4) Daca $D \le \dfrac{C_3}{C_2 - C_1}$, *atunci numarul optim de defecte detectate este m*=0 si timpul de livrare software optim este* $T^* = 0$.

2.1. Politici de eliberare software optimale din punct de vedere fiabilitate-cost

Decizia e conditionata de numarul optim de defecte care trebuie detectate astfel încât sa fie minimizat costul software total C(m), dat de ecuatia (I), astfel încât:

$$R(x_0;m) = \sum_{j=0}^{m} [m!/(m-j)!j!]\, p^j q^{m-j} * \exp(-Dk^j x_0)$$

sa satisfaca obiectivul de fiabilitate R_0.
Problema livrarii software optimala poate fi formulata astfel: pentru un timp operational specificat $x_0 (x_0 \ge 0)$, se urmaresc: minimizarea C(m), conditionat de $R(x_0;m) \ge R_0$,

$0<R_0<1$. Se defineste m_0, numar finit si unic care satisface $R(x_0;m)\geq R_0$ si $R(x_0;m-1)<R_0$, $1<m_0<\infty$. Aceasta problema este numita a livrarii software optimale în ceea ce priveste raportul fiabilitate-cost [3]. În acest sens, se formuleaza urmatoarea teorema:

Teorema 2. *Presupunem ca* $C_2\rangle C_1\rangle 0$, $C_3\rangle 0$, $x_0 \geq 0$ *si* $0\langle R_0\langle 1$.

(1) Daca $D\rangle \frac{C_3}{C_2-C_1} \geq \frac{D}{(p/k+q)^{M-1}}$ *si* $\exp(-Dx_0)\langle R_0$, *atunci numarul optimal de defecte detectate este* $m^*=\max\{m_0, M\}$ *si timpul de livrare software optim este:*
$T^* = \left|1-(p/k+q)^{m^*}\right| / pD(1-1/k)$;

(2) Daca $\frac{D}{(p/k+q)^{M-1}}\rangle \frac{C_3}{C_2-C_1}$ *si exp(-Dx0) <R0, atunci numarul optim de defecte detectate este* $m^*=\max\{m_0, M\}$ *si timpul de livrare software optim este:*
$T^* = \left|1-(p/k+q)^{m^*}\right| / pD(1-1/k)$;

(3) Daca $D\rangle \frac{C_3}{C_2-C_1} \geq \frac{D}{(p/k+q)^{M-1}}$ *si* $\exp(-Dx_0)\geq R_0$, *atunci numarul optim de defecte detectate este* $m^*=m_1$ *(vezi ec. III) si timpul de livrare software optim este:*
$T^* = \left|1-(p/k+q)^{m^*}\right| / pD(1-1/k)$;

(4) Daca, $\frac{D}{(p/k+q)^{M-1}}\rangle \frac{C_3}{C_2-C_1}$ *si exp(-Dx0) ³R0 atunci numarul optim de defecte detectate este* $m^*=M$ *si timpul de livrare software optim este:*
$T^* = \left|1-(p/k+q)^{m^*}\right| / pD(1-1/k)$;

(5) Daca, $D\leq \frac{C_3}{C_2-C_1}$ *si exp(-Dx0)<R0, atunci numarul optim de defecte detectate este* $m^*=m_0$ *si timpul de livrare software optim este:*
$T^* = \left|1-(p/k+q)^{m^*}\right| / pD(1-1/k)$;

(6) Daca, $D\leq \frac{C_3}{C_2-C_1}$ *si exp(-Dx0)³R0, atunci numarul optim de defecte detectate este* $m^*=0$ *si timpul de livrare software optim este:* $T^*=0$.

3. Concluzii

Programele de management al dezvoltarii software bine conduse presupun ca cerintele de baza program (cost, planificare, performanta si suport) sunt planificate si executate cu succes. Aceasta implica identificarea riscurilor si folosirea tehnicilor de micsorare a lor; estimarea costurilor reale; estimarea planificarii. Sprijinul calitatii produsului este acordat în permanenta si la un nivel înalt de competenta, în conformitate cu standardele acceptate în domeniu. Prin toate aceste masuri se urmareste predictibilitatea: cost predictibil, planificare predictibila, performanta si suport predictibile. Cu alte cuvinte, calitate predictibila.

Bibliografie

1. [FENT91] Fenton, N., Software Measurement: Why a formal approach?, in Proceedings of the BCS-FACS Workshop on Formal Aspects of Measurement, South Bank University, London, 5 May 1991, pg. 20-27.
2. [KITC91] Kitchenham, B., Never mind the metrics what about the numbers!, in Proceedings of the BCS-FACS Workshop on Formal Aspects of Measurement, South Bank University, London, 5 May 1991, pg. 29-37.
3. [YAMA95] Yamada, S., Tokuno, K., Osaki, S., Software Reliability Measurement in Imperfect Debugging Environment, in Software Reliability and Testing, IEEE Computer Society Press, Los Alamitos, California, 1995, pg. 66-73.
4. [***92] AC/317-D/10, ADP Manpower Estimation for Software Maintenance, Nato, Brussels, 1992.
5. [***96] MIL-HDBK-781A, Handbook for Reliability Test Methods, Plans, and Environments for Engineering, Development Qualification and Production, Departament of Defense, S.U.A., 1996.