Estimarea costului unui proiect software

9.1. Costuri şi efort

Estimarea costurilor unei aplicaţii software este greu de realizat cu precizie

- este presupusă o relaţie simplă între cost şi efort;
- efortul poate fi măsurat în luni-om.
- Există o relaţie între efortul necesar şi mărimea produsului (KLOC – kilolines of code, kilo-linii de cod).

Pentru a determina ecuaţiile unui model algoritmic de estimare a costului există mai multe abordări:

1. Un parametru variază în timp ce ceilalţi parametri rămân constanţi şi se determină influenţa parametrului variabil asupra rezultatului
    - De ex. comentariile asupra depanării şi întreţinerii
2. Se analizează datele unor proiecte anterioare
    - De ex. timpul necesar fazelor de dezvoltare, calificarea personalului implicat, mărimea produsului final;

Analiza costurilor permite identificarea unor strategii de creştere a productivităţii software:

- Scrierea de mai puţin cod
- Stimularea oamenilor să lucreze la capacitatea maximă
- Evitarea refacerii componentelor dezvoltate anterior
- Dezvoltarea şi folosirea mediilor de dezvoltare integrate

9.2. Modelul Halstead

Îşi propune o estimare mai obiectivă a mărimii unui program pe baza unui număr de unităţi sintactice: operanzi şi operatori

- Entităţi de bază:

  n_1 = numărul de operatori diferiţi

  n_2 = numărul de operanzi diferiţi

  N_1 = numărul total de apariţii ale operatorilor

  N_2 = numărul total de apariţii ale operanzilor

  Vocabularul: $n = n_1 + n_2$

  Lungimea implementării: $N = N_1 + N_2$

Ecuaţia lungimii: $N' = n_1 \log_2 n_1 + n_2 \log_2 n_2$

Volumul programului: $V = N \log_2 n$

- Variabile

- Estimare empirică a lungimii programului în linii de cod:

LOC = 102 + 5,31 * *VARS*

- Fiecare program care va conţine aproximativ 100 de linii de cod, plus 5 linii suplimentare pentru fiecare variabilă care apare în program.

Generalizarea acestor rezultate la programe mai mari nu este indicată.

În programele mai complexe, factori precum interfaţa dintre componente şi comunicarea necesară între persoanele implicate joacă un rol ce nu poate fi neglijat.

Într-o abordare naivă am putea considera că un proiect ce necesită 60 de luni-om poate fi realizat într-un an cu o echipă de 5 persoane sau într-o lună cu o echipă de 60 de persoane.

Această abordare însă este prea simplistă.

Costurile estimate au deseori o culoare politică, iar rezultatele sunt determinate de argumente care nu au o natură tehnică.

Dacă avem 12 luni pentru a finaliza o lucrare, ea va necesita 12 luni.

Acest motiv poate fi privit ca o variantă a *legii Parkinson*: munca ocupă tot timpul disponibil.

Dacă ştim că o ofertă de 100.000 de euro a fost făcută de concurenţă, noi vom face o ofertă de 90.000 de euro. Acesta este cunoscut sub denumirea de *preţ de câştig.*

Dorim să ne promovăm produsul la un anumit târg de tehnică de calcul şi din acest motiv programul trebuie scris şi testat în următoarele 9 luni, deşi realizăm că timpul este limitat.

Această situaţie este cunoscută sub denumirea de *metoda bugetului* de estimare a costului.

Proiectul poate fi dezvoltat într-un an, dar şeful nu ar accepta acest termen. Ştim că termenul de 10 luni este acceptabil şi atunci îl programăm pentru 10 luni.

Simpla comparare a caracteristicilor unui proiect cu un proiect precedent nu garantează o estimare corectă a costului său.

Dacă o echipă lucrează în mod repetat la proiecte asemănătoare, timpul de lucru necesar va scădea, datorită acumulării experienţei.

În 1968, unei echipe de programatori i s-a cerut să dezvolte un compilator FORTRAN pentru trei maşini diferite.

Rezultate:

Compilatorul	Efortul (în luni-om)
1	72
2	36
3	14

Consultarea experţilor

- Metoda Delphi

Fiecare expert îşi expune opinia în scris.

Un moderator colectează estimările obţinute astfel şi le redistribuie celorlalţi experţi.

Numele experţilor nu sunt asociate cu estimările lor.

Fiecare expert va preda o nouă estimare bazată pe informaţiile primite de la moderator.

Procesul continuă până când se ajunge la un consens

- Distribuţia beta

Un expert realizează mai multe estimări:

- estimare optimistă *a*,
- o estimare realistă *m* şi
- o estimare pesimistă *b*.

Efortul aşteptat va fi:

$$E = (a + 4m + b) / 6,$$

o estimare mai bună probabil decât dacă s-ar fi considerat numai media aritmetică a lui *a* şi *b.*

9.3.Modele algoritmice clasice - Modele liniare

Efortul pentru realizarea unui proiect este:

$$E = a_0 + \sum_{i=1}^{n} a_i x_i$$

- Coeficienţii a_i sunt constante, iar x_i reprezintă factorii care au impact asupra efortului necesar;
- *E* reprezintă efortul (de exemplu, numărul necesar estimat de luni-om);

Modelul Nelson

$$E = -33{,}63 + 9{,}15x_1 + 10{,}73x_2 + 0{,}51x_3 + 0{,}46x_4 + 0{,}40x_5 + 7{,}28x_6 - 21{,}45x_7$$
$$+ 13{,}5x_8 + 12{,}35x_9 + 58{,}82x_{10} + 30{,}61x_{11} + 29{,}55x_{12} + 0{,}54x_{13} - 25{,}20x_{14}$$

Factor	Descriere	Valori posibile
x_1	Instabilitatea specificaţiilor cerinţelor	0-2
x_2	Instabilitatea proiectării	0-3
x_3	Procentajul de instrucţiuni matematice	0-100
x_4	Procentajul de instrucţiuni I/O	0-100
x_5	Numărul subprogramelor	număr
x_6	Utilizarea unui limbaj de nivel înalt	0(da) / 1(nu)
x_7	Aplicaţie comercială	0(da) / 1(nu)
x_8	Program de sine stătător	0(da) / 1(nu)
x_9	Primul program pe această maşină	1(da) / 0(nu)
x_{10}	Dezvoltare concurentă de hardware	1(da) / 0(nu)
x_{11}	Utilizarea dispozitivelor random-access	1(da) / 0(nu)
x_{12}	Maşină gazdă diferită de maşina ţintă	1(da) / 0(nu)
x_{13}	Număr de erori	număr
x_{14}	Dezvoltare pentru o organizaţie militară	0(da) / 1(nu)

Dacă avem o estimare *E*, atunci efortul real *R* va verifica formula:

$$P((1-\alpha)E \le R \le (1+\alpha)E) \ge \beta,$$

unde valori acceptabile pentru α şi β sunt:

α = 0,2 şi β = 0,9.

Exemplu: Să presupunem că estimarea este de 100 luni-om.

Atunci probabilitatea ca proiectul să necesite în realitate între 80 şi 120 de luni-om este mai mare ca 90%.

Există o probabilitate diferită de zero ca efortul real să fie în afara intervalului.

Modelul Wolverton

Este o abordare bottom-up cu o matrice de costuri, care are un număr limitat de tipuri diferite de module şi un număr de nivele de complexitate.

Tipul modulului	Complexitate				
	Mică		↔		**Mare**
1. Management de date	11	13	15	18	22
2. Management de memorie	25	26	27	29	32
3. Algoritm	6	8	14	27	51
4. Interfaţă utilizator	13	16	19	23	29
5. Control	20	25	30	35	40

Fiind dată o matrice de costuri C, un modul de tip I, complexitate j şi mărime S_k, va rezulta un cost al modulului $M_k = S_k \cdot C_{ij}$

9.4. Modele algoritmice moderne – Modele neliniare

Forma generală este: $$E = (a + b \cdot KLOC^c) \cdot f(x_1, \ldots, x_n),$$

Valorile constantelor *a*, *b*, *c* rezultă pe baza unei analize de regresiune a datelor proiectelor disponibile. De obicei, *f* nu se ia în considerare.

Constanta *c*

Autor	Formula
Halstead	$E = 0.7 \cdot KLOC^{1.50}$
Boehm	$E = 2.4 \cdot KLOC^{1.05}$
Walston-Felix	$E = 5.2 \cdot KLOC^{0.91}$

- *c* < 1: analogie cu producţia de masă (costurile fixe se împart pe mai multe unităţi de produs);
- *c* > 1: efortul creşte exponenţial cu mărimea datorită creşterii complexităţii
  - Pentru proiecte mari, această relaţie pare mai plauzibilă.

Exemple

KLOC	Halstead	Boehm	Walston-Felix
1	0,7	2,4	5,2
10	22,1	26,9	42,3
50	247,5	145,9	182,8
100	700	302,1	343,6
1000	22135,9	3390,1	2792,6

Modelul Walston-Felix

A fost creat prin analiza a 60 de proiecte de la IBM.

Proiectele erau complet diferite ca mărime, iar programele au fost scrise în mai multe limbaje de programare.

Au fost identificate 29 de variabile care influenţează productivitatea.

Pentru fiecare din aceste variabile au fost considerate trei niveluri: mare, mediu şi mic.

Variabila	**Productivitatea medie pentru valoarea variabilei**			**PC**
Complexitatea interfeţei utilizator	< normală 500	normală 295	> normală 124	376
Calificarea şi experienţa personalului	mică 132	medie 257	mare 410	278
Experienţa anterioară cu aplicaţii similare	minimă 146	medie 221	vastă 410	264
Procentajul de programatori participanţi în faza de proiectare	< 25% 153	25 - 50% 242	> 50% 391	238
Raportul dintre mărimea medie a echipei şi durata proiectului (persoane/lună)	< 0,5 305	0,5 – 0,9 310	> 0,9 171	134

Walston şi Felix consideră că indexul productivităţii *I* poate fi determinat pentru noile proiecte după următoarea relaţie

$$I = \sum_{i=1}^{29} W_i X_i$$

unde ponderile W_i sunt definite astfel:

$$W_i = 0{,}5 \cdot \log_{10}(PC_i)$$

Modelul COCOMO

Modelul COCOMO – **CO**nstructive **CO**st **MO**del a lui Boehm este unul dintre cele mai cunoscute modele de cost care se aplică proiectelor.

Sunt trei clase de proiecte:

- *Organice:* o echipă relativ mică dezvoltă programul într-un mediu cunoscut. Sunt de obicei programe relativ mici.
- *Integrate:* sisteme pentru care mediul impune constrângeri severe (de ex. programe de control al traficului aerian sau aplicaţiile militare);
- *Semidetaşate:* o formă intermediară;

Exemple

Clasa de proiect	**b**	**c**
organică	2,4	1,05
semidetaşată	3,0	1,12
integrată	3,6	1,20

KLOC	organic	semidetaşat	integrat
1	2,4	3,0	3,6
10	26,9	39,6	57,1
50	145,9	239,4	392,9
100	302,1	521,3	904,2
1000	3390	6872	14333

Analiza punctelor funcţionale

Se bazează pe numărarea diferitelor structuri de date utilizate.

Este potrivită mai ales pentru aplicaţiile comerciale, în care structura datelor are o foarte mare importanţă.

Este mai puţin indicată pentru proiectele în care algoritmii joacă rolul dominant (de ex. compilatoarele sau aplicaţiile în timp real).

Analiza se bazează pe modalităţile în care diverşi utilizatori interacţionează cu aplicaţiile.

Se consideră că sistemul îndeplineşte cinci funcţii fundamentale:

- Funcţii referitoare la date
    1. Fişiere interne logice;
    2. Fişiere externe de interfaţă ;
- Funcţii tranzacţionale
    3. Intrări externe;
    4. Ieşiri externe;
    5. Cereri externe;

1. **Fişiere interne logice** – în engleză “Internal Logical Files”, FIL. Permit utilizatorilor să folosească datele pe care trebuie să le întreţină. De exemplu, un pilot poate introduce datele de navigare la un terminal din carlingă înainte de plecare. Datele sunt stocate într-un fişier şi pot fi modificate în timpul misiunii. Pilotul este deci responsabil pentru întreţinerea acestor date.
2. **Fişierele externe de interfaţă –** în engleză “External Interface Files”, FEI. Utilizatorul nu este responsabil pentru întreţinerea datelor; acestea sunt localizate în alt sistem care le întreţine.

Utilizatorul sistemului analizat solicită datele doar pentru informare. De exemplu, un pilot se poate informa asupra poziţiei cu ajutorul sateliţilor GPS sau al sistemelor de la sol. El nu are responsabilitatea actualizării acestor date, însă le poate accesa în timpul zborului.

3. **Intrările externe,** în engleză , “External Input”, IE. Permit utilizatorului să întreţină fişierele interne logice prin operaţii de adăugare, modificare şi ştergere.
4. **Ieşirile externe,** în engleză, “External Output”, EE. Permit utilizatorului să producă date de ieşire. De exemplu, pilotul poate să afişeze separat viteza la sol şi viteza reală în aer, informaţii derivate din datele interne (pe care le poate întreţine) şi cele externe (pe care le poate accesa).
5. **Cererile externe,** în engleză, “External Inquiries”, CE. Pentru ca utilizatorul să poată selecta şi afişa datele din fişiere, el trebuie să introducă informaţii de selecţie pentru a găsi datele în conformitate cu anumite criterii. În această situaţie datele din fişiere nu sunt modificate, ci doar căutate şi furnizate. De exemplu, dacă pilotul afişează date cu privire la relieful solului, date stocate anterior, rezultatul este regăsirea directă a informaţiilor.

Puncte funcţionale neajustate

Prin încercări repetate, s-au stabilit ponderi pentru fiecare dintre aceste entităţi. Numărul de puncte funcţionale neajustate este:

$$PFN = 10 \cdot FIL + 7 \cdot FEI + 4 \cdot IE + 5 \cdot EE + 4 \cdot CE$$

În funcţie de complexitatea tipurilor de date, se disting o serie de valori pentru aceste puncte funcţionale, prezentate în tabelul următor:

Tip	**Nivel de complexitate**		
	Simplu	Mediu	Complex
FIL	7	10	15
FEI	5	7	10
IE	3	4	6
EE	4	5	7
CE	3	4	6

Puncte funcţionale ajustate

Pentru ajustarea suplimentară a estimărilor, se iau în calcul şi alte 14 caracteristici care influenţează dezvoltarea aplicaţiilor:

- Comunicaţiile de date
- Funcţiile distribuite
- Performanţa
- Folosirea masivă a configuraţiilor
- Rata tranzacţiilor
- Intrările de date online
- Eficienţa utilizatorilor finali
- Actualizările online
- Prelucrările complexe
- Refolosirea
- Uşurinţa la instalare
- Uşurinţa la folosire
- Locaţiile multiple
- Facilitarea modificărilor

Influenţa fiecărei caracteristici este evaluată pe o scară de la 0 (nu influenţează) la 5 (influenţă puternică). Gradul de influenţare *GI* este suma acestor puncte pentru toate caracteristicile

- Se calculează apoi factorul de complexitate tehnică: *FCT* = 0,65 + 0,01 * *GI*
- Punctele funcţionale ajustate (*PF*) se obţin astfel: *PF = PFN * FCT.*

Avantaje

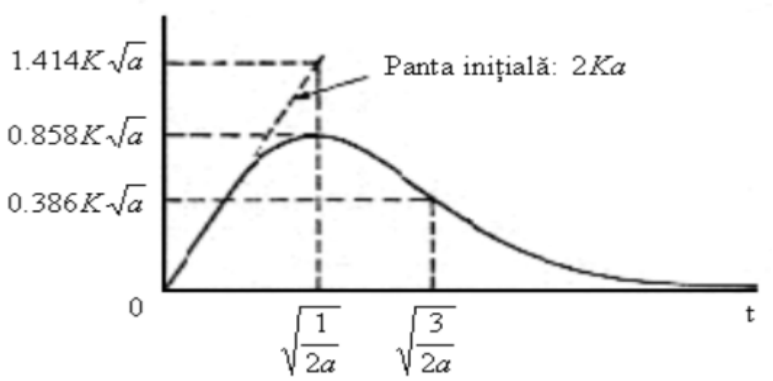
- Măsura productivităţii este un rezultat natural, deoarece punctele funcţionale sunt independente de tehnologie şi deci pot fi utilizate pentru a compara productivitatea pe platforme diferite şi cu instrumente de dezvoltare diferite;
- Ele pot fi folosite pentru a stabili o rată de productivitate (*PF* / *h*) care facilitează estimările privind durata proiectului ca întreg;

Distribuirea forţei de muncă în timp

Modelul Putnam-Norden

Distribuţia forţei de muncă în timp are de multe ori o formă caracteristică, bine aproximată de distribuţia Rayleigh. Astfel, forţa de muncă necesară la un moment de timp *t* este:

$$FM(t) = 2 \cdot K \cdot a \cdot t \cdot \mathrm{e}^{-at^2}$$



Maximul curbei este apropiat de momentul de timp în care proiectul va fi predat clientului.

Acest rezultat este foarte apropiat de o regulă euristică foarte des utilizată: 40% din efortul total este cheltuit pentru dezvoltarea efectivă, în timp ce 60% este cheltuit pentru întreţinere.

Ecuaţia software-ului

Putnam a folosit observaţii empirice legate de nivelurile de productivitate pentru a deriva *ecuaţia software-ului* din curba Rayleigh:

$$\boxed{D = k \cdot E^{1/3} \cdot t^{4/3}},$$

Unde D este dimensiunea proiectului, E este efortul total în ani-om, t este timpul scurs până la lansare în ani iar K este un factor tehnologic bazat pe 14 componente, precum:

- Maturitatea generală a proiectului şi tehnicile de management;
- Gradul de utilizare a tehnicilor de ingineria programării;
- Nivelul limbajelor de programare folosite;
- Capacitatea şi experienţa echipei de dezvoltare;
- Complexitatea aplicaţiei.

Efortul

Pentru estimarea efortului, Putnam a introdus *ecuaţia acumulării forţei de muncă:*

$$A = E/t^3,$$

unde A este numită accelerarea forţei de muncă iar E şi t au semnificaţiile de mai sus.

Accelerarea forţei de muncă este 12,3 pentru proiecte software noi, cu multe interfeţe şi interacţiuni cu alte sisteme, 15 pentru sisteme de sine stătătoare şi 27 pentru reimplementări ale sistemelor existente.

Pe baza celor două ecuaţii putem elimina timpul şi determina efortul:

$$E = (D/k)^{9/7} \cdot A^{4/7}$$

Acest rezultat este interesant deoarece arată că efortul este proporţional cu dimensiunea la puterea 9/7 ≈ 1,286, valoare similară cu factorul Boehm, între 1,05 şi 1,20.

Consecinţe

- Scurtarea timpului de dezvoltare implică un număr mai mare de persoane necesare pentru proiect;
- Referindu-ne la modelul curbei Rayleigh, scurtarea timpului de dezvoltare conduce la mărirea valorii *a*, factorul de accelerare care determină panta iniţială a curbei; vârful curbei Rayleigh se deplasează spre stânga şi în acelaşi timp în sus;
- Astfel obţinem o creştere a puterii necesare la începutul proiectului şi o forţă de muncă maximă mai mare.

Legea lui Brooks

Mai multe studii au arătat că productivitatea individuală scade odată cu creşterea echipei. Conform lui Brooks, există două cauze ale acestui fenomen:

- Creşte timpul acordat comunicării cu ceilalţi membri ai echipei (pentru consultare, sincronizarea sarcinilor etc.);
- Mai întâi scade productivitatea, deoarece noii membri ai echipei nu sunt productivi de la început. În acelaşi timp ei necesită ajutor, deci timp, de la ceilalţi membri ai echipei în timpul procesului de învăţare

- *Legea lui Brooks*: adăugarea de personal la un proiect întârziat îl va întârzia şi mai mult.

Mărimea echipei şi productivitatea

Productivitatea individuală (măsurată în linii de cod pe lună-om) scade cu mărimea echipei.

Mărimea echipei	Productivitatea individuală	Productivitate totală
1	500	500
2	450	900
3	400	1200
4	350	1400
5	300	1500
5,5	275	1512
6	250	1500
7	200	1400
8	1500	1200

Concluzii

- Nu există o relaţie simplă între preţul unui sistem şi costul său de dezvoltare;
- Factorii care afectează productivitatea includ aptitudinile individuale, experienţa în domeniu, natura proiectului, dimensiunea proiectului, instrumentele utilizate şi mediul de lucru;
- Preţul unui produs software poate fi stabilit astfel încât să se câştige un contract şi apoi funcţionalitatea este ajustată conform preţului;
- Pentru estimarea costului pot fi folosite tehnici diferite;
- Modelele algoritmice de estimare a costurilor se bazează pe analiza cantitativă a caracteristicilor proiectelor, permiţând compararea influenţei acestor caracteristici;
- Timpul necesar terminării unui proiect nu este proporţional cu numărul de persoane care lucrează la proiect.