UNIVERSITATEA TEHNICĂ "GHEORGHE ASACHI" DIN IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
AN UNIVERSITAR 2016-2017

Managementul Proiectelor Software

ASIST.ING. CORINA CÎMPANU

Conținutul cursului

- **Capitolul 1.** Introducere în managementul proiectelor
- **Capitolul 2.** Metodologii de dezvoltare a programelor
- **Capitolul 3.** Justificarea financiară a proiectului
- **Capitolul 4.** Managementul domeniului
- **Capitolul 5.** Managementul timpului
- **Capitolul 6.** Managementul costului
- **Capitolul 7.** Managementul calității
- **Capitolul 8.** Managementul resurselor umane
- **Capitolul 9.** Conducerea echipei
- **Capitolul 10.** Managementul comunicării
- **Capitolul 11.** Managementul riscului
- **Capitolul 12.** Analiza deciziilor

MANAGEMENTUL PROIECTELOR SOFTWARE

Cursul nr. 6

MANAGEMENTUL COSTULUI

Capitolul 6. Managementul costului

Estimarea costurilor unei aplicații software reprezintă un domeniu mai puțin formalizat, care se bazează mai mult pe aproximări. Există totuși o serie de metode care permit estimarea costului total pentru un proiect software pe baza unui număr limitat de generatori relevanți de costuri.

În cele mai multe modele de estimare, este presupusă o **relație simplă între cost şi efort**.

Efortul poate fi măsurat de exemplu în *luni-om* (adică numărul estimativ de luni necesar unui singur om să realizeze lucrarea), şi fiecărei luni-om i se asociază o sumă fixă de bani, corespunzător salariului angajaților.

Costul total estimat se obține înmulțind numărul estimat de luni-om cu factorul constant considerat.

Noțiunea de cost total reprezintă de obicei costul efortului inițial de dezvoltare software, costul analizei cerințelor, proiectării, implementării şi testării, fără a fi luate în considerare costurile de întreținere.

Noțiunea de cost, care se va folosi în continuare, nu include şi posibilele costuri hardware, ci numai costurile legate de personalul angajat în dezvoltarea produsului software.

Capitolul 6. Managementul costului
Metode de estimare

Cercetările în domeniul estimării costului sunt departe de a fi cristalizate. Diferite modele folosesc diferite sisteme de măsură şi generatoride costuri, încât este dificilă compararea lor. Să presupunem că un model foloseşte o ecuație de forma:

$$E = 2{,}7 * KLOC^{1{,}05}$$

Această ecuație arată o relație între efortul necesar şi mărimea produsului (*KLOC* = Kilo-Lines OfCode, kilo-linii de cod). Unitatea de măsură a efortului poate fi numărul de luni-om necesare.

Apar aici mai multe întrebări:

- Ce este o linie de cod?
- Numărăm codul maşină sau codul sursă într-un limbaj de nivel înalt?
- Numărăm de asemenea şi comentariile, liniile libere care măresc vizibilitatea sau nu?
- Luăm în considerare şi zilele libere, vacanțele, concediile medicale în noțiunea de luni-om sau se referă numai la o măsurare netă?

Diferite interpretări ale acestor întrebări pot conduce la rezultate complet diferite.Uneori nici nu este cunoscut ce definiții au fost folosite în aplicarea modelului.

Capitolul 6. Managementul costului
Metode de estimare

În mod ideal, mărimea ar trebui să se refere doar la codul propriuzis, fără linii albe şi comentarii, însă pentru proiecte de dimensiuni mari, este mai simplu să se numere direct toate liniile. Pentru a determina ecuațiile unui model algoritmic de estimare a costului, putem urma diferite abordări.

În primul rând ne putem baza ecuația pe rezultatul experimentelor. Într-un asemenea experiment, în general variem un parametru, în timp ce ceilalți parametri rămân constanți. În acest fel încercăm să determinăm influența pe care o are parametrul care variază.

Un exemplu tipic este cel ce testează dacă comentariile ajută la înțelegerea unui program. Vom pune un număr de întrebări despre acelaşi program unor programatori împărțiți în două grupuri. Primul grup va primi programul fără comentarii, iar al doilea grup va primi textul aceluiaşi program, dar comentat. Folosind rezultatele celor două grupuri, putem testa ipoteza realistă că o înțelegere mai bună şi mai rapidă a programului are efecte pozitive asupra întreținerii programului.

Capitolul 6. Managementul costului
Metode de estimare

O altă modalitate de a descoperi modele algoritmice de estimare a costului se bazează pe o analiză a datelor unui proiect real, dar şi pe un suport teoretic. O organizaţie poate strânge date despre un număr de sisteme software care au fost dezvoltate. Aceste date pot privi timpul petrecut la diferite faze bine determinate, calificarea personalului implicat, momentele de timp în care au apărut erori, atât în timpul testării cât şi după instalare, complexitatea, robusteţea şi alţi factori importanţi ai proiectului, mărimea diferitelor entităţi implicate şi o analiză statistică a acestor date.

Un exemplu pentru o asemenea relaţie este cea dată mai sus, ce exprimă o legătură între *E* şi *KLOC*.

Aplicabilitatea şi corectitudinea acestor ecuaţii este, în mod evident, dependentă de corectitudinea datelor pe care se bazează. Rezultatele obţinute în acest fel reprezintă o medie, o aproximare bazată pe datele disponibile, de aceea rezultatele obţinute trebuie aplicate cu atenţie.

De exemplu, programul ce urmează a fi dezvoltat în cadrul unui nou proiect nu se poate compara cu produse anterioare datorită inovaţiilor implicate. Estimarea costurilor pentru un proiect legat de o navetă spaţială nu poate fi făcută printr-o simplă extrapolare a proiectelor anterioare.

Trebuie reţinut că aplicarea oarbă a unor formule din modelele existente nu va rezolva problema estimării costului. Fiecare model necesită o adaptare la mediul în care va fi folosit. Aceasta conduce la necesitatea colectării continue a datelor din propriul proiect şi aplicarea unor metode statistice pentru a calibra parametrii modelului.

Capitolul 6. Managementul costului
Metode de estimare

Neconcordanţele dintre diferitele modele mai pot apărea deoarece:

- Majoritatea modelelor oferă o relaţie între numărul lunilor-om necesare şi mărimea produsului în linii de cod. Pot exista însă mai multe interpretări diferite pentru aceste noţiuni;
- Efortul nu înseamnă întotdeauna acelaşi lucru. Uneori se consideră activităţile de dezvoltare pornind de la conceperea produsului, după ce au fost fixate cerinţele. Alteori se includ şi eforturile de întreţinere.

Cu toate acestea, modelele de estimare a costului au multe caracteristici comune. Acestea reflectă importanţa factorilor care intervin asupra costului de dezvoltare şi efortului. Creşterea înţelegerii costurilor produselor software ne permite să identificăm **strategii de creştere a productivităţii**, cele mai importante fiind:

- ***Scrierea unei cantităţi mici de cod:*** Mărimea sistemului este una din cauzele principale ale efortului şi costului. Prin metode care încearcă să reducă mărimea, cum ar fi utilizarea de limbaje de nivel înalt, reutilizarea componentelor sau refactorizarea, se pot obţine reduceri semnificative;
- ***Stimularea oamenilor să lucreze la capacitatea maximă:*** Capacitatea de a lucra atât individual cât şi în echipă are un mare impact asupra productivităţii. Angajarea celor mai buni oameni este de obicei o afacere profitabilă. O mai bună stimulare, condiţii mai bune de lucru, cursurile de perfecţionare asigură oportunităţi de creştere a productivităţii;
- ***Evitarea rescrierii componentelor dezvoltate anterior:*** Studiile au arătat că pentru a rescrie ceea ce s-a produs deja este necesar un efort considerabil. Prin aplicarea unor tehnici precum prototipizarea, se pot reduce semnificativ costurile;
- ***Dezvoltarea şi folosirea mediilor de dezvoltare integrate,*** cu instrumentele ce pot ajuta la eliminarea sau eficientizarea unor etape.

Capitolul 6. Managementul costului
Metode de estimare – Modelul Halstead

Numeroase studii au fost efectuate cu scopul de a estima efortul necesar pentru activitățile elementare de programare.

Primele experimente au fost realizate de Halstead (1977).

- La baza modelului său stă faptul că **numărarea liniilor de cod poate constitui o problemă**, chiar dacă avem o definiție exactă a termenului „linie de cod".
- Unele linii sunt mult mai complicate decât altele.
- Conform teoriei lui Halstead, **este mai bine să se pornească de la un număr de unități sintactice**, după cum sunt recunoscute de compilator.
- Halstead face distincția între operatori și operanzi. Operatorii denotă o operație; exemple sunt operatorii standard (+, –, * etc.), dar și semnul de punctuație punct și virgulă (;) care arată structura unei instrucțiuni și construcții cum ar fi *if-then-else* și *while-do*. Operanzii înseamnă date: variabile și constante.
- Calcularea numărului de operatori și operanzi dintr-un program va oferi o măsură mai bună a mărimii decât simpla calculare a numărului de linii.

Capitolul 6. Managementul costului
Metode de estimare – Modelul Halstead (2)

În modelul Halstead, cele **patru entități de bază** pentru un program sunt:

- *n*1 = numărul de operatori diferiți;
- *n*2 = numărul de operanzi diferiți;
- *N*1 = numărul total de apariții ale operatorilor;
- *N*2 = numărul total de apariții ale operanzilor;

Relațiile modelului Halstead sunt următoarele:

- Vocabularul: $n = n_1 + n_2$
- Lungimea implementării: $N = N_1 + N_2$
- Ecuația lungimii: $N' = n_1 \log_2 n_1 + n_2 \log_2 n_2$
- Volumul programului: $V = N \log_2 n$
- Dificultatea: $D = \frac{n_1}{2} \cdot \frac{N_2}{n_2}$
- Efortul: $E = D \cdot V$

Capitolul 6. Managementul costului
Metode de estimare – Modelul Halstead (3)

Valoarea lui N depinde de n_1 și n_2.

Valoarea lui n_1 este un factor constant pentru programele scrise într-un anumit limbaj de nivel înalt și depinde de limbajul de programare ales.

O ipoteză consideră că n_2 este determinat în principal de numărul de variabile VARS care apar în program

$$\boldsymbol{LOC = 102 + 5.31 \cdot VARS}$$

Generalizarea acestor rezultate pentru programe mai mari nu este indicată!

Capitolul 6. Managementul costului
Metode de estimare – Modelul Halstead (3)

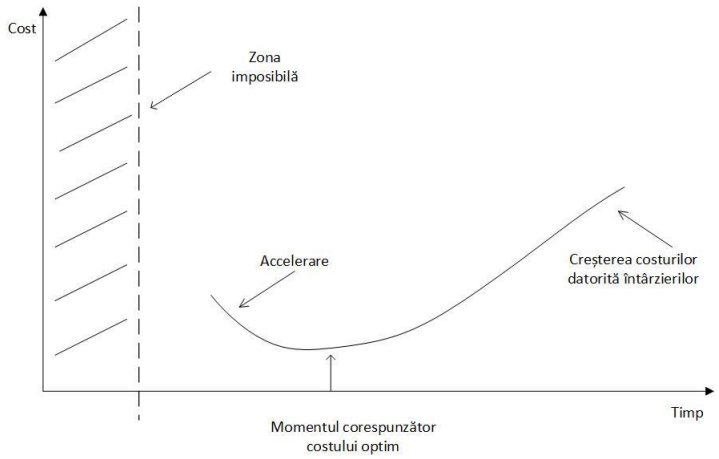
Exemplu:

Fie un proiect care necesită pentru realizare:

E = 60 luni-om

T = 1 an, echipă de 5 persoane

T = 1 lună, echipă de 60 de persoane



Capitolul 6. Managementul costului
Metode de estimare – Modelul Halstead (3)

Motive tipice care reflectă aceste argumente non-tehnice sunt prezentate în exemplele următoare:

- 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 a fost făcută o ofertă de 100.000 de euro de către concurență, noi vom face o ofertă de 90.000 de euro (**metoda *prețului 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;
- 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.

Capitolul 6. Managementul costului
Metode de estimare – Modelul Halstead (3)

Aceste estimări pot avea efecte negative, după cum s-a demonstrat frecvent în istoria ingineriei programării. Estimările vor depinde mai mult de argumentele politice ale părților interesate decât de realitatea tehnică a proiectului.

Pe de altă parte, 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 scade, datorită experienței acumulate. În 1968, unei echipe de programatori i s-a cerut să dezvolte un compilator FORTRAN pentru trei maşini diferite. Efortul necesar pentru aceste trei proiecte este descris în tabelul de mai jos:

Compilatorul	Efortul [luni-om]
1	72
2	36
3	14

Capitolul 6. Managementul costului
Metode de estimare – Metoda Delphi

Pentru estimarea costului se poate apela la serviciul unui expert, care apelează la propria sa experiență. Factori greu de cuantificat, precum caracteristicile de personalitate sau unele caracteristici neobişnuite ale proiectului, pot fi astfel luați în considerare. În acest caz, calitatea estimării nu poate depăşi calitatea expertului.

Pentru o estimare mai precisă se pot solicita mai mulți experți. Totuşi, dacă un grup de persoane trebuie să găsească împreună o soluție, se observă că unii membri ai grupului au un impact mai mare asupra rezultatului decât ceilalți. Unele persoane nu îşi exprimă opinia sau sunt impresionate de celelalte. Acest lucru poate avea un impact negativ asupra rezultatului final.

Pentru a anticipa acest efect negativ, putem folosi ***metoda Delphi.***

În metoda Delphi, fiecare expert îşi expune opinia în scris. Un moderator colectează estimările obținute astfel şi le redistribuie celorlalți experți. În acest proces nu sunt asociate numele experților cu estimările lor.

Fiecare expert predă apoi o nouă estimare bazată pe informațiile primite de la moderator. Procesul continuă până când se ajunge la un consens.

Capitolul 6. Managementul costului
Metode de estimare – Metoda estimărilor multiple

O altă metodă pentru obținerea unei estimări mai bune se bazează pe un expert care să realizeze mai multe estimări:

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

Efortul aşteptat va fi:

$$E = \frac{a + 4m + b}{6}$$

Această estimare este mai bună decât dacă se consideră numai media aritmetică a lui *a* şi *b*.

Capitolul 6. Managementul costului
Modele algoritmice clasice – Modelul Nelson

Nelson (1966) oferă un model liniar pentru estimarea efortului necesar pentru un proiect de dezvoltare software. Modelele liniare au următoarea formă:

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

Coeficienții a_i sunt constante, iar x_i reprezintă factorii care au impact asupra efortului necesar. Un număr mare de factori poate influența productivitatea şi implicit efortul. Analizând cu atenție datele proiectelor precedente şi diferite combinații de factori, se poate încerca obținerea unui model cu un număr mic de factori. **Nelson, de exemplu, sugerează un model care ia în considerare 14 factori:**

$$E = -33.63 + 9.15x_1 + 10.73x_2 + 0.51x_3 + 0.46x_4 + 0.40x_5 + 7.28x_6 - 21.45x_7 + 13.5x_8 + 12.35x_9 + 58.82x_{10} + 30.61x_{11} + 29.55x_{12} + 0.54x_{13} - 25.20x_{14}$$

În această ecuație E reprezintă numărul estimat de luni-om necesare. Seminificația factorilor x_i şi domeniile lor de definiție sunt prezentate în tabelul următor:

Capitolul 6. Managementul costului
Modele algoritmice clasice – Modelul Nelson(2)

Factor	Descriere	Valori posibile
x_1	Instabilitateaspecificațiilorcerințelor	0-2
x_2	Instabilitatea proiectării	0-3
x_3	Procentajul de instrucțiuni matematice	0-100
x_4	Procentajul de instrucțiuni I/O	0-100
x_5	Numărul subprogramelor	Număr
x_6	Utilizarea unui limbaj de nivel înalt	0(da)/ 1(nu)
x_7	Aplicație comercială	0(da)/ 1(nu)
x_8	Program de sine stătător	0(da)/ 1(nu)
x_9	Primul program pe această mașină	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}	Mașină gazdă diferită de mașina țintă	1(da)/ 0(nu)
x_{13}	Număr de erori	Număr
x_{14}	Dezvoltare pentru o organizație militară	0(da)/ 1(nu)

Capitolul 6. Managementul costului
Modele algoritmice clasice – Modelul Nelson(3)

Putem face mai multe observații asupra acestui model.

De exemplu, la dezvoltarea programelor pentru aplicațiile de apărare, în care programele sunt încorporate în mașini diferite de mașina gazdă, cum ar fi programul de control al zborului pentru rachete, factori precum x_{12} și x_{14} au cu siguranță un impact mare asupra costului.

Un alt exemplu este creşterea efortului atunci când se foloseşte limbajul de asamblare în locul unui limbaj de nivel înalt(x_6).

În general modelele liniare nu funcționează foarte bine. Deşi există un număr mare de factori care influențează productivitatea, este puțin probabil ca ei să intervină în mod independent şi liniar.

Trebuie atrasă atenția asupra preciziei acestui tip de formulă şi asupra capcanei exprimată în sloganul: **există trei tipuri de minciuni: minciuni mici, minciuni mari şi statistici.**

Formula lui Nelson este rezultatul analizei statistice a datelor unor proiecte reale şi trebuie interpretată ca atare, adică în termeni probabilistici.

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

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

Capitolul 6. Managementul costului
Modele algoritmice clasice – Modelul Nelson(4)

Să presupunem că $\alpha = 0{,}2$ și $\beta = 0{,}9$, iar estimarea este de 100 luni-om.

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

- Costurile estimate prin acest tip de model rezultă într-un anumit interval, rămânând o probabilitate diferită de zero ca acesta să fie în afara intervalului.
- Aplicabilitatea acestor estimări este puternic influențată de mărimea intervalului şi de probabilitatea ca valoarea reală a costului să aparțină într-adevăr acelui interval.
- În special pentru proiectele care necesită efort mai mare, este bine să se considere valoarea superioară a intervalului în care se află costul, în locul valorii estimate.

Capitolul 6. Managementul costului
Modele algoritmice clasice – Modelul Wolverton

O altă modalitate prin care un expert poate ajunge la **o estimare a costului este printr-un proces bottom-up**.

Pentru fiecare modul, se obține un cost estimativ iar costul final este suma costurilor modulelor, cu o corecție aplicată datorită integrării modulelor.

Wolverton descrie un model în care **o matrice de costuri este folosită ca punct de plecare în determinarea costurilor modulelor. În această matrice există un număr limitat de tipuri diferite de module şi un număr de niveluri de complexitate**. Tabelul următor ilustrează o matrice ipotetică de costuri.

Elementele matricei reflectă costul pentru fiecare linie de cod.

Capitolul 6. Managementul costului
Modele algoritmice clasice – Modelul Wolverton (2)

Tipul modulului	Complexitate				
	Mică		<- ->		Mare
	1	2	3	4	5
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. Interfață utilizator	13	16	19	23	29
5. Control	20	25	30	35	40

Capitolul 6. Managementul costului
Modele algoritmice clasice – Modelul Wolverton (2)

Fiind dată o matrice de costuri C, un modul de tip *i*, complexitate *j* şi mărime S_k, rezultă un cost al modulului *k*,

$$M_k = S_k \cdot C_{ij}$$

Acest tip de model are la rândul său unele probleme:

- Utilizatorul trebuie să estimeze subiectiv clasa de complexitate din care face parte fiecare modul, ceea ce determină un grad mare de nesiguranță.
- Alți factori care au un impact asupra productivității, cum ar fi experiența în programare şi caracteristicile hardware, nu sunt luați în considerare.
- Extinderea matricei pentru a include şi aceşti factori ar creşte gradul de subiectivitate al metodei.

Capitolul 6. Managementul costului
Modele algoritmice moderne

În secțiunile anterioare am remarcat faptul că efortul de programare este corelat cu mărimea programului. **Există şi modele neliniare care exprimă această legătură**. O formă generală este:

$$E = (a + b \cdot KLOC^C) \cdot f(x_1, \dots, x_n)$$

unde:

- *KLOC* – reprezintă mărimea programului în kilo-linii de cod;
- *E* – reprezintă efortul în luni-om;
- *a, b* şi *c* – sunt constante;
- $f(x_1, \dots, x_n)$ – este o funcție care depinde de valorile factorilor.

În general, **formula de bază** este:

$$E = (a + b \cdot KLOC^C)$$

Capitolul 6. Managementul costului
Modele algoritmice moderne

Aceasta este obținută printr-o analiză de regresie a datelor proiectelor disponibile.

Primul generator de cost este mărimea programului, măsurată în linii de cod.

Acest cost nominal estimat este apoi adaptat pentru un număr de factori care influențează productivitatea (generatorii de cost secundari).

De exemplu, dacă unul din factorii folosiți reprezintă „experiența echipei de programatori" aceasta poate cauza o corecție a costului nominal estimat cu 1,50, 1,20, 1,00, 0,80, 0,60 pentru un nivel de expertiză foarte scăzut, scăzut, mediu, înalt şi respectiv foarte înalt.

Tabelul următor conține formulele de bază pentru relația dintre mărimea programului şi efort. Din motivele enunțate anterior este dificil să comparăm aceste modele. Este interesant de observat că valoarea lui c variază în jurul valorii de 1 în toate modelele.

Capitolul 6. Managementul costului
Modele algoritmice moderne

Tabelul următor conține formulele de bază pentru relația dintre mărimea programului şi efort. Din motivele enunțate anterior este dificil să comparăm aceste modele. Este interesant de observat că valoarea lui c variază în jurul valorii de 1 în toate modelele.

Modelul	Formula
Halstead	$E = 0.7 \cdot KLOC^{1,50}$
Boehm	$E = 2.4 \cdot KLOC^{1,05}$
Walston-Felix	$E = 5.2 \cdot KLOC^{0,91}$

Capitolul 6. Managementul costului
Modele algoritmice moderne

Când c < 1, se remarcă apariția unui fenomen foarte bine cunoscut din teoria economică. Pentru producția de masă, se presupune că este mai ieftin să se producă mari cantități din acelaşi produs. Costurile fixe vor fi împărțite astfel unui număr mai mare de unități, ceea ce conduce la scăderea costului pe unitate. În cazul programelor, liniile de cod sunt produsul, deci presupunem că producând multe linii de cod, scade costul pe linie de cod. Motivul este costul instrumentelor scumpe precum mediile de dezvoltare, instrumentele de testare sau generatoarele de programe, care poate fi distribuit unui număr mai mare de linii de cod.

În cazul opus, când c > 1, observăm că după un anumit punct, producerea de unități suplimentare implică nişte costuri suplimentare. Programele foarte mari sunt mult mai scumpe, deoarece creşte necesitatea de comunicare şi de control managerial, iar problemele şi interfețele sunt mai complexe. Deci fiecare linie de cod suplimentară necesită mai mult efort. **Al doilea tip de relații (c > 1) pare mai plauzibil. Pentru proiecte foarte mari, efortul creşte mai mult decât liniar cu mărimea.**

Alegerea unui anumit model depinde în principal de gradul de complexitate a interfațării modulelor proiectului. Este evident că valoarea exponentului c influențează foarte mult valoarea calculată *E*, în special pentru valori mari ale *KLOC*.

Capitolul 6. Managementul costului
Modele algoritmice moderne

Tabelul următor prezintă valorile lui *E*, calculate prin metodele prezentate anterior şi pentru câteva valori ale *KLOC*. Se remarcă marea diferență dintre modele.

Pentru programe mici, prin metoda Halstead rezultă costuri estimate mici.

Pentru proiecte cu aproximativ un milion de linii de cod, acelaşi model generează estimări ale costului cu un ordin de mărime mai mari decât prin aplicarea metodei Walston-Felix.

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

Capitolul 6. Managementul costului
Modele algoritmice moderne

Aceste observații nu trebuie să ne conducă la concluzia că aceste metode sunt inutile. Există diferențe mari între caracteristicile mulțimilor de proiecte pe care se bazează diferite modele. Ştim că numerele utilizate în modele provin în urma analizei datelor proiectelor reale. Dacă aceste date reflectă diferite proiecte şi/sau medii de dezvoltare, modelele se vor comporta la fel. Nu putem copia pur şi simplu aceste formule. Fiecare mediu are caracteristicile sale proprii şi este deci necesară adaptarea parametrilor la mediul specific, proces numit *calibrare*.

Cea mai importantă problemă este obținerea unei estimări inițiale a mărimii programului.

- Cum am putea să estimăm numărul de pagini ale unui roman care nu a fost scris încă?
- Chiar dacă ştim numărul de personaje, de locații şi intervalul în care se va desfăşura povestea, este dificilă estimarea realistă a mărimii încă de la început.
- Cu cât înaintăm în realizarea proiectului, cu atât va fi mai exactă estimarea mărimii.
- Dacă proiectarea se apropie de final, ne putem forma o impresie rezonabilă asupra mărimii programului rezultat.
- Dar numai când sistemul va fi predat vom cunoaşte valoarea exactă.

Capitolul 6. Managementul costului
Modelul Walston-Felix

Ecuația ce stă la baza **modelului Walston-Felix (1977)** este:

$$E = 5,2 \cdot KLOC^{0,91}$$

Modelul a fost creat prin analiza a 60 de proiecte de la IBM. Aceste proiecte erau complet diferite ca mărime, iar programele au fost scrise în mai multe limbaje de programare. Totuşi nu reprezintă o surpriză faptul că dacă aplicăm acest model chiar unei submulțimi a celor 60 de proiecte, nu vom avea rezultate satisfăcătoare.

Încercând să explice aceste rezultate dintr-o plajă mare de valori, autorii au identificat 29 de variabile care influențează în mod sigur productivitatea. Pentru fiecare din aceste variabile au fost considerate trei niveluri: mare, mediu şi mic.

Pentru un număr de 51 de proiecte, Walston şi Felix au determinat nivelul fiecărei variabile din cele 29, împreună cu productivitatea obținută, exprimată ca număr de linii de cod pe lună-om.

Aceste rezultate sunt prezentate în tabelul următor pentru câteva din cele mai importante variabile. De exemplu, productivitatea medie este de 500 de linii de cod pe lună-om pentru proiecte cu o interfață utilizator de complexitate scăzută. Pentru o interfață de complexitate înaltă sau medie, productivitatea este de 295 şi respectiv 124 de linii de cod pe lună. **Ultima coloană reprezintă variația productivității, *PC* (engl. "productivity change"), diferența dintre valorile maxime şi minime**.

Capitolul 6. Managementul costului
Modelul Walston-Felix

Variabila	Productivitatea medie pentru valoarea variabilei			Productivity Change PC=maxVal-minVal
Complexitatea interfeței utilizator	<Normală 500	Normală 295	>Normală 124	376
Calificarea și experiența personalului	Mică 132	Medie 257	Mare 410	278
Experiență anetrioară cu aplicații similare	Minimă 146	Medie 221	Vastă 410	264
Procentajul de programatori	<25% 153	25-50% 242	>50% 391	238
Raportul dintre mărimea medie a echipei și durata proiectului (persoane/lună)	<0.5 305	0,5-0,9 310	>0.9 171	134

Capitolul 6. Managementul costului
Modelul Walston-Felix

Walston și Felix consideră **că indexul productivității I** poate fi determminat pentru noile proiecte după următoarea relație:

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

unde ponderile W_i sunt definite astfel:

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

În relația de mai sus PC_i reprezintă variația productivității factorului i.

Pentru primul factor din tabelul de mai sus, complexitatea interfeței cu utilizatorul rezultă următoarele: $PC_1 = 376$, deci $W_1 = 1,29$.

Variabilele X_i pot lua valorile -1,0 și 1, unde factorul corespunzător este de nivel scăzut, mediu sau înalt. Indexul productivității obținut poate fi tradus într-o productivitate așteptată: linii de cod scrise pe lună-om.

Modelul pleacă de la ecuația de bază și folosește I pentru a ajusta estimările.

- Numărul factorilor luați în considerare în acest model este destul de ridicat: 29 de factori din 51 de proiecte.
- De asemenea, nu este clar cum se influențează acești factori unul pe celălalt. Un alt dezavantaj ar fi că numărul de alternative pentru fiecare factor este de numai 3 și nu oferă destule opțiuni pentru situațiile practice.

Capitolul 6. Managementul costului
Modelul COCOMO [Boehm]

COCOMO (engl. "COnstructive COşt MOdel") este una dintre cele mai bine documentate metode de estimare a costului (Boehm, 1981). În forma sa cea mai simplă, numită Basic COCOMO, formula care exprimă legătura dintre efort şi mărimea programului, este:

$$E = b \cdot KLOC^c$$

unde b şi c sunt constante ce depind de tipul proiectului executat.

Boehm distinge trei clase de proiecte:

- **Organice** – în proiectele de acest tip o echipă relativ mică dezvoltă produsul într-un mediu cunoscut. Persoanele implicate pot să lucreze de la început, fiind necesare investiţii iniţiale. Proiectele de acest tip sunt de multe ori programe relativ mici;
- **Integrate** – proiectele de acest tip implică sisteme unde mediul impune constrângeri severe. Produsul va fi integrat într-un mediu foarte strict. Exemple de asemenea proiecte sunt programele de control al traficului aerian sau aplicaţiile militare;
- **Semidetaşate** – această clasă de proiecte reprezintă o formă intermediară. Echipa poate fi formată atât din persoane experimentate cât şi neexperimentate, proiectul poate fi destul de mare, dar nu foarte mare.

Capitolul 6. Managementul costului
Modelul COCOMO

- Pentru clase diferite, parametrii metodei *Basic COCOMO* iau următoarele valori:

Clasa de proiect	b	C
Organică	2,4	1,05
Semidetaşată	3,0	1,12
integrată	3,6	1,20

Tabelul următoar prezintă estimări ale efortului pentru fiecare din cele trei moduri, pentru diferite valori ale KLOC (deşi un proiect organic de un milion de linii de cod nu este realist). Se observă influenţa foarte mare a constantei c asupra estimărilor obţinute. Estimările efortului sunt exprimate tot în luni-om.

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		521,3	904,2
1000		6872	14333

Capitolul 6. Managementul costului
Analiza punctelor funcționale

***Analiza punctelor funcționale* (Albrecht, 1983)** este o metodă de estimare a costurilor care încearcă să evite problemele determinate de estimarea dimensiunii codului.

- APF se bazează pe numărarea diferitelor structuri de date utilizate. Se presupune că acest număr este un bun indicator pentru dimensiunea proiectului.
- Metoda este potrivită mai ales pentru aplicațiile comerciale în care structura datelor are o foarte mare importanță.
- APF este mai puțin indicată pentru proiectele în care algoritmii joacă rolul dominant, de exemplu compilatoarele sau aplicațiile de timp real.
- Unul din scopurile *principale* ale APF este evaluarea sistemului din punctul de vedere al utilizatorilor.
- De aceea, analiza se bazează pe modalitățile în care diverşi utilizatori interacționează cu aplicațiile.

Astfel, sistemul îndeplineşte cinci funcții fundamentale:

- ***funcții referitoare la date:***
  - fişiere interne logice;
  - fişiere externe de interfață;
- ***funcții tranzacționale:***
  - intrări externe;
  - ieşiri externe;
  - interogări externe.

Capitolul 6. Managementul costului
Analiza punctelor funcționale

***Fişierele interne logice* (engl. "Internal Logical Files", *ILF*): Permit utilizatorilor să folosească datele pe care trebuie să le întrețină.**

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 întreținerea acestor date.

***Fişierele externe de interfață* (engl. "External Interface Files", *EIF*): În acest caz, utilizatorul nu este responsabil pentru întreținerea datelor; acestea sunt localizate în alt sistem care le întreține. Utilizatorul sistemului analizat solicită datele doar pentru informare.**

De exemplu: un pilot se poate informa asupra poziției cu ajutorul sateliților GPS sau al sistemelor de la sol. El nu are responsabilitatea actualizării acestor date, însă le poate accesa în timpul zborului.

Capitolul 6. Managementul costului
Analiza punctelor funcționale

***Intrările externe* (engl. "External Input", *EI*): Permit utilizatorului să întrețină fişierele interne logice prin operații de adăugare, modificare şi ştergere.**

***Ieşirile externe* (engl. "External Output", *EO*): Permit utilizatorului să producă date de ieşire.**

De exemplu, pilotul poate să afişeze separat viteza la sol şi viteza reală în aer, informații derivate din datele interne (pe care le poate întreține) şi cele externe (pe care le poate accesa).

***Interogările externe* (engl. "External Inquiries", *EQ*): Pentru ca utilizatorul să poată selecta şi afişa datele din fişiere, el trebuie să introducă informații de selecție pentru a găsi datele în conformitate cu anumite criterii. Interogările pot accesa date atât din fişierele interne logice cât şi din fişierele externe de interfață, dar nu produc date noi. 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 informațiilor.

Capitolul 6. Managementul costului
Analiza punctelor funcționale

Prin încercări repetate, s-au stabilit ponderi pentru fiecare din aceste entități.

Numărul de puncte funcționale neajustate este:

$$PFN = 10 \cdot ILF + 7 \cdot EIF + 4 \cdot EI + 5 \cdot EO + 4 \cdot EQ$$

În funcție 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
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EO	4	5	7
EQ	3	4	6

Capitolul 6. Managementul costului Analiza punctelor funcționale

Pentru ajustarea suplimentară a estimărilor se iau în calcul și alte 14 caracteristici care influențează dezvoltarea aplicațiilor: comunicațiile de date, funcțiile distribuite, performanța, folosirea masivă a configurațiilor, rata tranzacțiilor, intrările de date online, eficiența utilizatorilor finali, actualizări online, prelucrările complexe, refolosirea, ușurința la instașare, ușurința la folosire, locațiile multiple, facilitarea modificărilor. **Influența fiecărei caracteristici este evaluată la o scară de la 0(nu influențează) la 5(influență puternică). Gradul de influențare GI este suma acestor puncte pentru toate caracteristicile.** Se calculează apoi **factorul de complexitate tehnică:**

$$\mathbf{FCT} = \mathbf{0,65} + \mathbf{0,01} \cdot \mathbf{GI}$$

Punctele funcționale ajustate (PF) se obțin astfel:

$$\mathbf{PF} = \mathbf{PFN} \cdot \mathbf{FCT}$$

- Avantajul principal al analizei punctelor funcționale este faptul că măsura productivității este un rezultat natural deoarece puntele funcționale sunt independente de tehnologie și deci pot fi utilizate pentru a compara productivitatea pe platforme diferite și 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.

Capitolul 6. Managementul costului Distribuirea forței de muncă în timp [Norden]

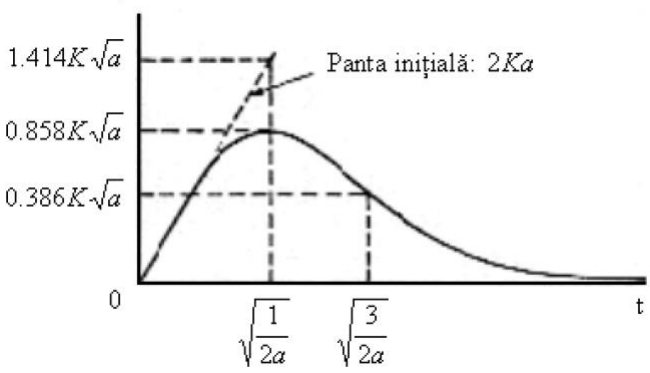
Norden a studiat distribuția forței de muncă în timp într-un număr de proiecte de dezvoltare software din anii '60. A descoperit că această distribuție avea deseori o formă caracteristică, bine aproximată de distribuția Rayleigh. Bazându-se pe această descoperire, Putnam a dezvoltat un model de estimare a costului în care **forța de muncă necesară la un moment de timp t** este dată de relația:

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

unde:

- a este un factor de accelerare care determină panta inițială a curbei;
- K reprezintă forța de muncă totală necesară, incluzând faza de întreținere; K este egal cu aria zonei determinate de curba Rayleigh, reprezentată în figura următoare:

Capitolul 6. Managementul costului
Distribuirea forței de muncă în timp



Integrarea ecuației pentru $FM(t)$ determină **efortul cumulat *I:***

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

Dacă vom considera momentul de timp T în care curba Rayleigh ajunge în punctul de maxim, atunci

$$a = \frac{1}{2 \cdot T^2}$$

Acest moment este apropiat de momentul de timp în care proiectul este predat clientului. **Aria delimitată de curba Rayleigh între punctele 0 și T este o bună aproximare a efortului inițial de dezvoltare**. Pentru acesta obținem:

$$\boldsymbol{E = I(t) = 0,3945 \cdot K}$$

Capitolul 6. Managementul costului
Distribuirea forței de muncă în timp

Acest rezultat este foarte apropiat de o regulă euristică foarte des utilizată: 40% din efortul total este consumat pentru dezvoltarea efectivă, în timp ce 60% este consumat pentru întreținere. Specificarea cerințelor nu este inclusă în model și deci estimările nu se pot aplică decât începând cu proiectarea și implementarea.

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

$$\boldsymbol{D = k \cdot E^{\frac{1}{3}} \cdot T^{\frac{4}{3}}}$$

unde:

D – este dimensiunea proiectului;

E – este efortul total în ani-om;

t – este timpul scurs până la lansare[ani];

K – este un factor tehnologic bazat pe 14 componente, precum:

- maturitatea generală a proiectului și tehnicile de management;
- gradul de utilizare a tehnicilor de ingineria programării;
- nivelul limbajelor de programare folosite;
- capacitatea și experiența echipei de dezvoltare;
- complexitatea aplicației.

Capitolul 6. Managementul costului
Distribuirea forței de muncă în timp

Puterea supraunitară a timpului are implicații puternice asupra alocării resurselor în proiecte mari. Prelungiri relativ mici ale datei de livrare pot determina reducerea substanțială a efortului. Pentru estimarea efortului, Putnam a introdus ***ecuația acumulării forței de muncă (eng. "manpower-buildup")***:

$$A = \frac{E}{t^3}, \text{ unde:}$$

- *A este accelerarea forței de muncă;*
- *E este efortul total [ani-om];*
- T este timpul scurs până la lansare [ani].

Accelerarea forței de muncă este:

- 12,3 pentru proiecte software noi, cu multe interfețe şi interacțiuni cu alte sisteme;
- 15 pentru sisteme de sine stătătoare;
- 27 pentru reimplementări ale sistemelor existente.

Pe baza celor două ecuații eliminăm timpul și determinăm efortul:

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

Acest rezultat este interesant deoarece arată că efortul este proporțional cu dimensiunea la puterea 9/ 7, ≈1,286 , valoare similară cu factorul *c al lui Boehm c=1,20*.

Capitolul 6. Managementul costului
Distribuirea forței de muncă în timp

Evident, 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 inițială a curbei. Vârful curbei Rayleigh se deplasează spre stânga sus. Astfel obținem o creștere a puterii necesare la începutul proiectului și o forță de muncă maximă mai mare.

Există și **dezavantaje ale acestei deplasări**. Mai multe studii au arătat că **productivitatea individuală scade odată cu creșterea echipei.** Conform lui Brooks, există **două cauze** ale acestui fenomen:

- Dacă o echipă se mărește, crește timpul acordat comunicării cu ceilalți membri ai echipei (pentru consultare, sincronizarea sarcinilor, etc.);
- Dacă este adăugată forță de muncă suplimentară unei echipe în timpul dezvoltării unui proiect, mai întâi scade productivitatea. Noii membri ai echipei nu sunt productivi de la început, când necesită ajutor, deci timp, de la ceilalți membri ai echipei în timpul procesului de învățare. Luate împreună, acestea conduc la scăderea productivității totale.

Capitolul 6. Managementul costului
Distribuirea forței de muncă în timp

Combinând aceste două informații, ajungem la fenomenul cunoscut sub denumirea de legea lui Brooks: adăugarea de personal la un proiect întârziat îl va întârzia şi mai mult. Analizând o mare bază d edate de proiecte, Conte a descoperit următoarea relație între productivitatea medie L (măsurată în linii de cod pe lună-om) și mărimea medie a unei echipe P:

$$L = \frac{777}{\sqrt{P}}$$

Formula atestă faptul că productivitatea individuală scade cu mărimea echipei. Numărul de legături de comunicare între persoanele implicate într-un proiect este determinat de mărimea și structura echipei.

Dacă într-o echipă de mărime P fiecare membru trebuie să-și coordoneze activitățile sale cu toți ceilalți din echipă, numărul legăturilor de comunicație va fi:

$$\frac{P(P-1)}{2}$$

Dacă fiecare membru trebuie să comunice numai cu un singur alt membru, acest număr va fi $P-1$. Mai puțină comunicare decât aceasta nu este rezonabilă, deoarece ne-am confrunta cu echipe independente.

Capitolul 6. Managementul costului
Distribuirea forței de muncă în timp

Numărul de legături de comunicație variază de la aproximativ P la aproximativ $P^2/2$. Într-o organizare ierarhică, aceasta conduce la P^{α} căi de comunicație, cu $\alpha\epsilon(1,2)$.

Pentru un membru al echipei, numărul de legături de comunicație variază de la 1 la P-1.Dacă productivitatea individuală maximă este L și fiecare legătură de comunicație conduce la o pierdere a productivității l, atunci productivitatea medie va fi:

$$L_\gamma = L - l \cdot (P-1)^\gamma$$

unde $\gamma \in (0,1]$ este o măsură a numărului de legături de comunicație.

Presupunem că există cel puțin o persoană care să comunice cu mai mult de o persoană, deci $\gamma > 0$. Pentru o chipă de mărime P, aceasta conduce la o productivitate totală:

$$L_{tot} = P \cdot L_\gamma = P \cdot (L - l \cdot (P-1)^\gamma)$$

Capitolul 6. Managementul costului
Distribuirea forței de muncă în timp

Pentru o mulțime dată de valori L, l și γ, pentru valori crescătoare ale lui P, această funcție crește de la 0 la valoarea maximă și apoi scade din nou. Deci, există o mărime optimă a echipei P_{opt}, care conduce la o productivitate maximă a echipei. Productivitatea echipei pentru diferite valori ale lui P este dată în tabelul următor. Aici presupunem că productivitatea individuală este de 500LOC/lună-om (L=500), iar scăderea de productivitate este de 10% pe canal de comunicație ($l = 5$). Cu interacțiune completă între membrii echipei $\gamma = 1$, rezultă o echipă optimă de 5,5 persoane.

Mărimea echipei	Productivitatea individuală	Productivitatea 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	150	1200

Capitolul 6. Managementul costului
Concluzii

În acest curs au fost prezentate diferite modele de estimare a efortului necesar pentru dezvoltarea unui proiect software, a foței de muncă și a timpului efectiv de dezvoltare.

În final, s-a prezentat o modalitate de estimare a distribuției forței de muncă în timp pentru a identifica numărul optim de persoane implicate într-un proiect.