**Tema de casa**

Costul estimativ al produselor software

prof. coord.: Stefan Stancescu

student: Costoae Eduard-Mihail

grupa: 441A – diferenta (2013-2014)

Cuprins

Introducere 3

Metode de estimare. 5

Metode algoritmice Clasice 7

Metode algoritmice moderne 8

Bibliografie 13

1. Introducere

Estimarea costurilor unui produs software este un domeniu mai putin cunoscut si se bazeaza in general pe aproximari. Exista totusi o serie de metode carepermit estimarea costului total pentru un proiect software pe baza unui numar limitat de generatori relevanti de costuri. În cele mai multe modele de estimare, este presupusa o relatie simpl a între cost si efort. Efortul poate fi masurat de exemplu în *luni-om* (adica numarul estimativ de luni necesar unui singur om sa realizeze lucrarea), si fiecarei luni-om i se asociaza o suma fixa de bani, corespunzator salariului angajatilor.

Costul total estimat se obtine înmultind numarul estimat de luni -om cu factorul constant considerat. Notiunea de cost total reprezinta de obicei costul efortului initial de dezvoltare software, costul analizei cerintelor, proiectarii, implementarii si testarii, fara a fi luate în considerare costurile de întretinere. Notiunea de cost, care se va folosi în continuare, nu include si posibilele costuri hardware, ci numai costurile legate de personalul angajat în dezvoltarea produsului software.

Estimarea costurilor software-ului este procesul de prezicere a efortului necesar pentru a dezvolta un produs software. Au fost propuse de-a lungul ultimilor 30 de ani multe modele de estimare. Această lucrare oferă o prezentare generală a metodelor de estimare a costului produselor software,inclusiv recentele progrese în domeniu. Unele din aceste modele se bazează pe o estimare a dimensiunii software-ului de intrare. Se va

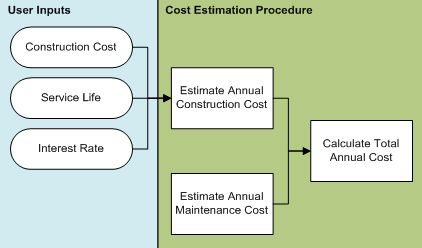
prezenta mai întâi o imagine de ansamblu a metricilor de dimensiune uzuale. Se vor evidenția mai apoi modelele de estimare de cost care au fost propuse și utilizat e cu succes. Modelele pot fi clasificate în două mari categorii: algoritmice și nonalgoritmice.

Fiecare model are propriile puncte forte și puncte slabe. Un factor-cheie în selectarea unui model de estimare a costuli este acuratețea estimărilor sale. Din păcate, în ciuda corpului mare de experiența cu modele de estimare, acuratețea acestor modele nu este satisfăcătoare.

Estimarea resurselor necesare și a timpului sunt foarte importante pentru toate proiectele, dar mai ales în cadrul sistemelor informatice, în cazul în care bugetul și programul sunt de obicei depășite.Toate metodele de estimare trebuie să se ia ca referință o dimensiune metrica a software-ului. Estimarea proiectelor software prezinta dificultati speciale, în comparație cu alte sectoare.

Metodele de estimare existente sunt extrem de dependente de informatiile disponibile pentru proiect. Când proiectul este impartit in mai multe etape,estimarea este mai precisă, deoarece există mai multe informații și este mai fiabil. Procesul de estimare ar trebui să fie un proces continuu, incluzand noi informații.Acest lucru stabileste o metodă de estimare de costuri ale software-ului bazată pe tehnici de inteligență artificială (AI), identificând factorii care influențează cel mai mult și cei variabili in costul software-ului. Munca se face pe baza unui set de date din proiectele trecute, luând datele furnizate de International Software

Benchmarking Standards Group (ISBSG), care a colectat informații de la mai mult de 2000 de proiecte. Acest set de date conține valori numerice, precum și date categoriale.



In imaginea de mai sus avem o schema a metodologiei estimarii costului produselor software.

1. Metode de estimare.

Diferite modele folosesc diferite sisteme de masura si generatori de costuri, încât este dificila compararea lor. Sa presupunem ca un model foloseate o ecuatie de forma:

E = 2.7 \* KLOC^1.05

Acesta ecuatie arata o relatie între efortul necesar si marimea produsului (KLOC= KiloLinesOf Code, kilo-linii de cod). Unitatea de masura a efortului poate fi numarul de luni -om necesare. Pentru a determina ecuatiile unui model algoritmic de estimare a costului, putem urma diferite abordari. În primul rând ne putem baza ecuatia pe rezultatul experimentelor. Într -un asemenea experiment, în general variem unparametru, în timp ce ceilalti parametri ramân constanti.

În acest fel încercam sa determinam influenta pe care o are parametrul care variaza . Un exemplu tipic este cel ce testeaza daca comentariile ajuta la întelegerea unui program. Vom pune un numar de întrebari despre acelasi program unor programatori împartiti în doua grupuri.

Primul grup va primi programul fara comentarii, iar al doilea grup va primi textul aceluiasi program, dar comentat. Folosind rezultatele celor doua grupuri, putem testa ipoteza realista ca o întelegere mai buna si mai rapida a programului are efecte pozitive asupra întretinerii programului. O alta modalitate de a de scoperi modele algoritmice de estimare a costului se bazeaza pe o analiza a datelor unui proiect real, dar si pe un suport teoretic.

O organizatie poate strânge date despre un numar de sisteme software care au fost dezvoltate. Aceste date pot privi timpul petrecut la diferite faze

Metode de estimare non algoritmice

Există două tipuri majore de metode de estimare a costurilor: algoritmice și nonalgoritmice. Modele algoritmice variază foarte mult în complexitate matematică. Unele sunt bazate pe formule matematice simple, folosind astfel de statistici sumare ca medii și deviațiile standard.

Altele se bazează pe modele de regresie și ecuații diferențiale.Pentru a îmbunătăți precizia de modele algoritmice, există o nevoie de a ajusta saucalibra modelul de la circumstanțele locale.[1]

Metodele nealgoritmice sunt:

1. analogia costului
2. hotararea expertilor
3. Parkinson
4. Price-to-win
5. Bottom-up
6. Top-down

a. Analogia costului:

Această metodă necesită unul sau mai multe proiecte finalizate anterior, care sunt similare cu noul proiect și estimarea derivă prin analogie, folosind costurile efective ale proiectelor anterioare. Estimarea prin analogie se poate face fie la nivel total al proiectului sau la nivel de subsistem. Nivelul total al proiectului are avantajul că toate componentele de cost ale sistemului vor fi luate în considerare în timp ce nivelul subsistemului are avantajul de a oferi o evaluare mai detaliată a asemănărilor și deosebirilor dintre noul proiect și proiectele finalizate.

Punctul forte al acestei metode este că estimarea se bazează pe experiența reală a proiectului. Cu toate acestea, nu este clar în ce măsură proiectul anterior este de fapt reprezentant al constrângerilor, mediu și funcțiile care urmează să fie efectuate de către noul sistem.

b. Hotararea expertilor:

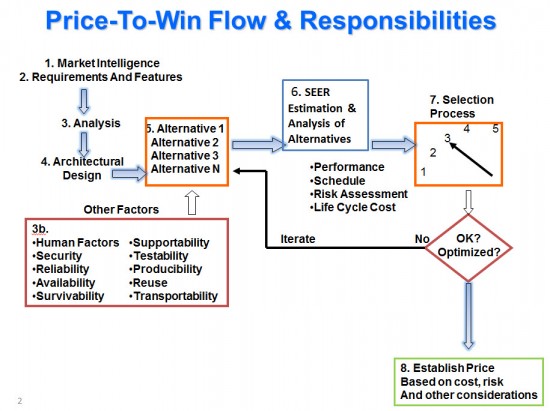
Această metodă implică consultarea unui sau mai multi experți. Experții oferă estimări utilizând propriile metode și experiența lor proprie. Mecanismele expertconsens, cum ar fi tehnica Delphi sau PERT vor fi folosite pentru a rezolva incoerențele în estimări .

c. Parkinson:

Utilizarea principiul Parkinson "munca se extinde pentru a umple volumul disponibil" costul este determinat (nu estimat) de resursele disponibile, mai degrabă decât bazandu-se pe o evaluare obiectivă. În cazul în care software-ul trebuie să fie livrate în 12 luni și 5 persoane sunt disponibile, efortul este estimat a fi 60 persoaneluni. Deși, uneori, această metodă nu este recomandată, deoarece poatefurniza estimări foarte nerealiste. De asemenea, această metodă nu promovează bunele practici în ingineria software.[2]

d. Price-to-win:

Costul programului este estimat a fi cel mai bun preț pentru a câștiga proiectul. Estimarea se bazează pe bugetul clientului în loc sa se bazeze pe funcționalitatea software. De exemplu, dacă o estimare rezonabilă pentru un proiect costă 100 de persoane-luni, însă clientul isi poate permite numai 60 persoane-luni, estimatorul este rugat să modifice estimarea efortului pentru a se potrivi 60 persoane-luni , în scopul de a câștiga proiectul . Acest lucru nu este din nou o bună practică, deoarece este foarte probabil sa se cauzeze o întârziere de livrare sau echipa de dezvoltare să lucreze ore suplimentare. [3]



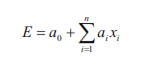
e. Bottom-up:

În această abordare, fiecare componentă a sistemului de software este separat estimată și rezultatele agregate pentru a produce o estimare pentru sistemul global. Cerința pentru această abordare este că un designul inițial trebuie să fie în locul care indică modul în care sistemul este descompus în componente diferite. [4]

1. Metode algoritmice Clasice

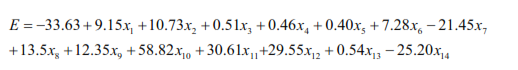
Nelson (1966) ofera un model liniar pentru estimarea efortului necesar pentru un proiect de dezvoltare software.

Modelele liniare au urmatoarea forma:

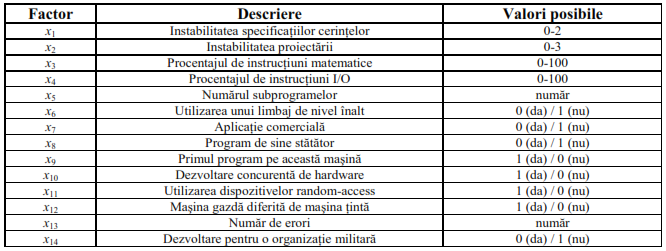


Coeficientii ai sunt constante, iar xi reprezinta factorii care au impact asupra efortului necesar. Un numar mare de factori poate influenta productivitatea si implicit efortul necesar. Analizând cu atentie datele proiectelor precedente si diferite combinatii de factori, putem încerca sa obtinem un model cu un numar mic de factori.

Nelson, de exemplu, sugereaza un model care ia în considerare 14 factori:



In aceasta ecuatie E reprezinta numarul estimat de luni-om necesar. Semnificatia factorilor x si domeniul lor de definitie sunt prezentate în tabelul urmator:



Putem face mai multe observatii asupra acestui model. In dezvoltarea programelor pentru aplicatiile de aparare, în care programele vor fi încorporate în masini diferite de masina gazda (un exemplu ar putea fi programul de control al zborului pentru rachete), factori precum x vor avea cu siguranta un impact mare asupra costului. Aceasta nu se va mai verifica însa într-un caz complet diferit.

Observam din nou ca datele proiectelor care stau la baza modelului au un impact semnificativ asupra factorilor care influenteaza costul. Exista o crestere a efortului datorita folosirii limbajului de asamblare în locul unui limbaj de nivel înalt (x6).

1. Metode algoritmice moderne

In paragrafele anterioare am remarcat faptul ca efortul de programare este corelat cu marimea programului. Exista si modele neliniare care exprima aceasta legatura. O forma generala este:

C:\Users\Eddy\Desktop\is5.png

unde KLOC reprezinta marimea programului în kilo-linii de cod, iar E reprezinta efortul în luni-om. a, b si c sunt constante iar f(,...,) este o functie care depinde de valorile factorilor x1,...,xn

In general, formula de baza este:

C:\Users\Eddy\Desktop\is6.png

Ea este obtinuta printr-o analiza de regresiune a datelor proiectelor disponibile. Primul generator de cost este marimea programului, masurata în linii de cod. Acest cost nominal estimateste apoi adaptat prin corectarea sa pentru un numar de factori care influenteaza productivitatea (generatorii de cost secundari).

De exemplu, daca unul din factorii folositi reprezinta experienta echipei de programatori aceasta poate cauza o corectie a costului nominal estimat cu 1,50, 1,20,1,00, 0,80, 0,60 pentru un nivel de expertiza foarte scazut, scazut, mediu, înalt si respectiv foarte înalt.

Modelul Walston-Felix.

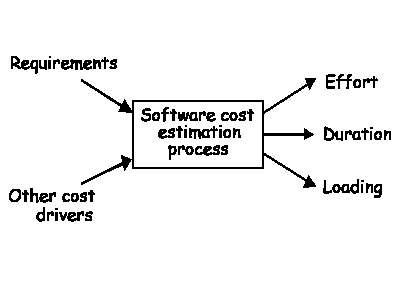
Ecuatia ce sta la baza modelului Walston-Felix este:

C:\Users\Eddy\Desktop\is7.png

Acest model a fost creat prin analiza a 60 de proiecte de la IBM. Aceste proiecte erau complet diferite ca marime, iar programele au fost scrise în mai multe limbaje de programare. Totusi nu reprezinta o surpriza faptul ca daca aplicam acest model chiar unei submultimi a celor 60 de proiecte, nu vom avea rezultate satisfacatoare.

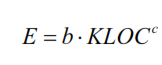
Incercând sa explice aceste rezultate dintr-o plaja mare de valori, Walston si Felix au identificat 29 de variabile care influenteaza în mod sigur productivitatea. Pentru fiecare din aceste variabile au fost considerate trei nivele: mare, mediu si mic. Pentru un numar de 51 de proiecte, Walston si Felix au determinat nivelul fiecarei variabile din cele 29, împreuna cu productivitatea obtinuta, exprimata ca numarul liniilor de cod pe luna-om.

Aceste rezultate sunt prezentate în tabelul urmator pentru câteva din cele mai importante variabile. De exemplu, productivitatea medie este de 500 de linii de cod pe luna-om pentru proiecte cu o interfata utilizator de complexitate scazuta. Pentru o interfata de complexitate înalta sau medie, productivitatea este de 295 si respectiv 124 de linii de cod pe luna. Ultima coloana reprezinta variatia productivitatii, PC (engl. productivity change), valoarea absoluta a diferentei dintre valorile minime si maxime.[5]



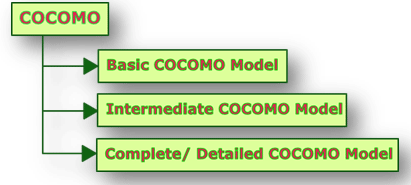
Modelul COCOMO

COCOMO (COnstuctive COst MOdel) este unul din cei mai bine documentati algoritmi de estimare a costului (Boehm, 1981). In forma sa cea mai simpla, numita Basic COCOMO, formula care exprima legatura dintre efort si marimea programului este:

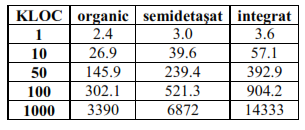


,unde b si c sunt constante ce depind de tipul proiectului ce este executat. Boehm distinge trei clase de proiecte:

* Organice: In proiectele de tip organic o echipa relativ mica dezvolta programul într-un mediu cunoscut. Persoanele implicate au în general experienta în proiecte similare realizateîn organizatia lor. Astfel, ei pot sa lucreze de la început, nefiind necesare investitii initiale.Proiectele de acest tip sunt de multe ori programe relativ mici;
* Integrate: Proiectele din acest tip implica sisteme unde mediul impune constrângeri severe.Produsul va fi integrat într-un mediu care este foarte strict. Exemplu de asemenea proiecte sunt programe de control al traficului aerian sau aplicatiile militare;
* Semidetasate: Aceasta este o forma intermediara. Echipa poate fi formata din persoane experimentate si neexperimentate, proiectul poate fi destul de mare, dar nu foarte mare.



Pentru clase diferite, parametrii metodei Basic COCOMO iau urmatoarele valori:



Tabelul urmator prezinta estimari ale efortului pentru fiecare din cele trei moduri, pentru diferite valori ale KLOC (desi un proiect organic de un milion de linii de cod nu este realist). Se observa influenta foarte mare a constantei c asupra estimarilor obtinute. Estimarile efortului sunt exprimate tot în luni-om.[6]

Concluzii finale

Estimarea costului unui proiect software este un concept simplu, insa este complex si dificil in realitate.Complexitatea necesara succesului depaseste capabilitatea multor manageri de proiect, insa este un aspect critic in dezvoltarea software. Multe proiecte au fost oprite din cauza rezultatelor nefavorabile ale estimarilor costului.

De aceea, acest aspect necesita depunerea de eforturi si schimbul de experienta cu cercetatori academici astfel incat companiile dezvoltatoare de software sa obtina modele de dezvoltare cat mai putin

costisitoare. Cea mai buna metoda de estimare a costului unui proiect software este folosirea unei combinatii intre unelte pentru estimarea costului, unelte pentru managementul proiectelor, sub stricta supraveghere a unor manageri de proiect experimentati si specialist in estimare.

1. Bibliografie
2. <http://www.ecfc.u-net.com/cost/models.htm>
3. <http://ifs.host.cs.st-andrews.ac.uk/Books/SE7/SampleChapters/ch26.pdf>
4. <http://www2.engr.arizona.edu/~ece473/readings/16-Emprical%20Validation.pdf>
5. <http://driverlayer.com/img/software%20cost%20estimation/147/any>
6. Software Estimation, Steve McConnell 2006
7. <http://pmbook.ce.cmu.edu/05_Cost_Estimation.html>

Imaginile folosite in lucrarea de fata nu se afla in proprietatea mea si sunt imprumutate din linkurile de mai sus.

6. Referinte

[1] Algorithmic Models for Estimating Software Effort, Ridhika Sharma

[2] Estimation techiniques, <http://www.cs.odu.edu/~price/cs451/Lectures/04mgmt/costest/costest_htsu3.html>  
[3] Price To Win: What It Is And When Estimation Gets Involved, Dan Galorath - http://www.galorath.com/wp/price-to-win-what-it-is-and-when-estimation-gets-involved.php

[4] Top down and bottom-up unit cost estimation, <http://data.gov.uk/sib_knowledge_box/top-down-and-bottom-unit-cost-estimation>

[5] Parametric Estimation of Software Systems, Kavita Choudhary

<http://www.ijsce.org/attachments/File/Vol-1_Issue-2/A024031211.pdf>

[6] COCOMO, Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

https://files.ifi.uzh.ch/rerg/arvo/courses/seminar\_ws02/reports/Seminar\_4.pdf