Procena troškova

6. Procena troškova

a) Mere troškova i faktori koji utiču na procenu troškova

b) Pristupi (osnovni metodi)

c) Metod “funkcijskih poena” (FP)

d) Primer primene FP metoda

e) Model konstruktivnih troškova (COCOMO)

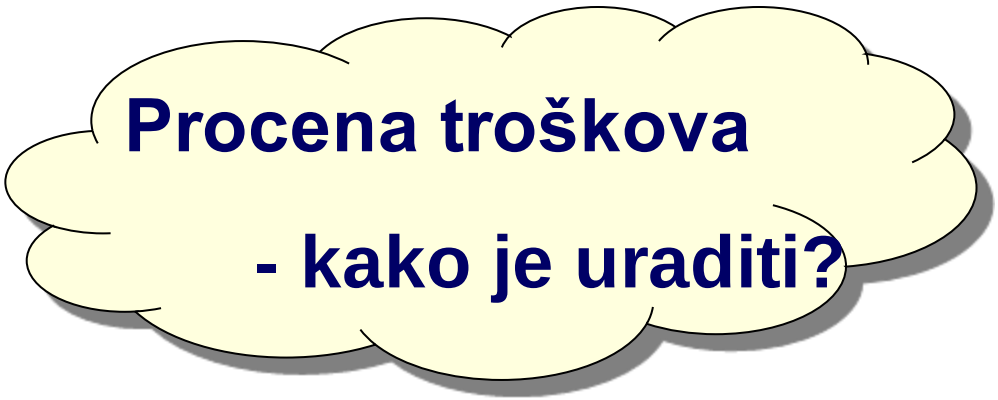
Motivacija

Zašto je potrebna procena troškova?

planiranje projekta: datumi, ljudstvo

Ugovori sa fiksnim cenama

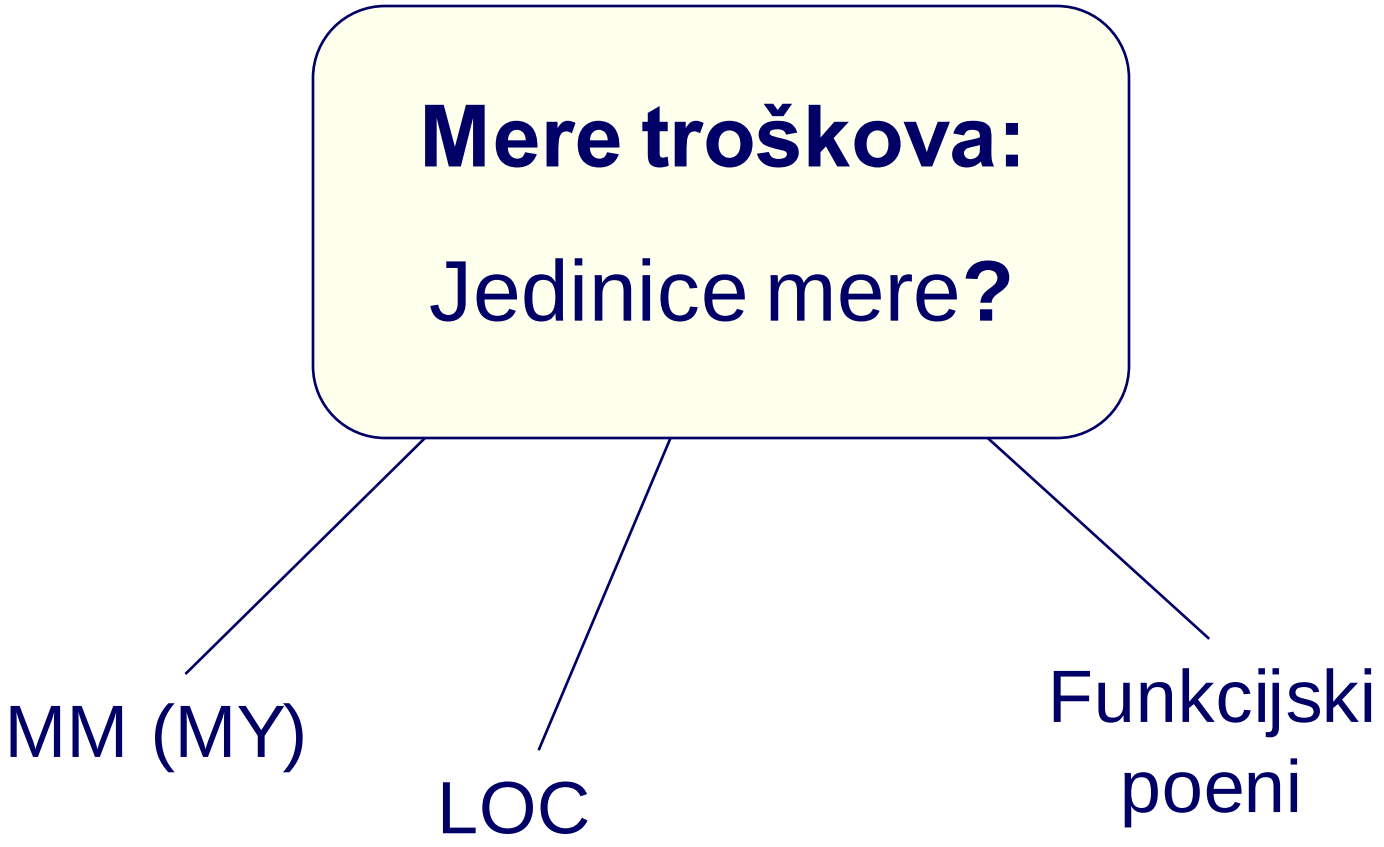
Početne ideje …?



Pitanja:

- Na kojim osnovama treba vršiti procenu (Input)?
- Kakvu vrstu rezultata treba očekivati (Output)?

Jedinice mere za procenu troškova



Jedinice mere: MM, MY

- Koliko vremena će trajati razvoj?
  - npr. 10 godina za jednog zaposlenog = 10 MY
    = 12*10 MM (man-month)
    = 5 godina za 2 zaposlenih
    = 1 godina za 10 zaposlenih
    = 1 dan za … zaposlenih … ??

- ali: problemi kounikacije rastu sa rastućim brojem članova tima!

Optimalno vreme razvoja softvera

Optimalno vreme razvoja softvera (cf. Boehm 1981)

$= 2{,}5 * (\text{trud u MM})^{S}$

(npr. S = 0,35 za online sisteme,
S = 0,32 za sisteme u realnom vremenu)

Primer:

- Procenjeni razvojni troškovi: 9 MM
- Optimalno trajanje = $2{,}5 * 9^{0{,}35} = 5{,}3$ meseci
- Broj zaposlenih
  = 9 MM / 5,3 meseci = 1,7 (tj. 2)

Linije izvornog koda

- Lines of code (LOC), ili
- Thousand delivered source instructions (KDSI)
  - Izvorni kod je samo jedan deo softverskog proizvoda
  - Različiti jezici $\Rightarrow$ različite dužine koda za istu funkcionalnost
  - Postoje nejasnoće oko brojanja linija koda
    - Izvršne linije?
    - Deklaracije podataka?
    - Komentari?
    - Komandni skriptovi (make i sl)?
    - Izmenjene/obrisane linije?
  - Mušterija ne dobija sve što se napiše

Linije izvornog koda (2)

- LOC je poznat tek pošto se proizvod završi
- Procena zasnovana na LOC nije sigurna
  - Za potrebe procene, LOC završenog proizvoda se mora proceniti
  - Procena LOC dovodi do procene troška projekta

Ocena produktivnosti zasnovana na LOC

- Što je jezik nižeg nivoa, sledi da je programer produktivniji
  - Ista funkcionalnost zahteva više koda na jeziku nižeg nivoa nego na jeziku višeg nivoa
- Što je programer “opširniji”, sledi da je i produktivniji
  - Merenje produktivnosti linijima koda dovelo bi do zaključka da je programer koji piše opširniji kod produktivniji od programera koji piše kompaktan kod.
- Zaključak: LOC je neadekvatan za procenu produktivnosti

Primer koji ovo ilustruje

Razvoj softvera identične funkcionalnosti na asembleru i na jeziku višeg nivoa

	Analiza	Dizajn	Implem	Test	Dokumentovanje
Asembler	3 nedelje	5 ned	8 ned	10 ned	2 nedelje
Viši prog. jezik	3 nedelje	5 ned	4 ned	6 ned	2 nedelje

	Veličina	Vreme	Produktivnost
Asembler	5000 linija	28 nedelja	714 linija/mesec
Viši prog. jezik	1500 linija	20 nedelja	300 linija/mesec

Faktori koji utiču na procenu troškova

- Od čega zavise troškovi?

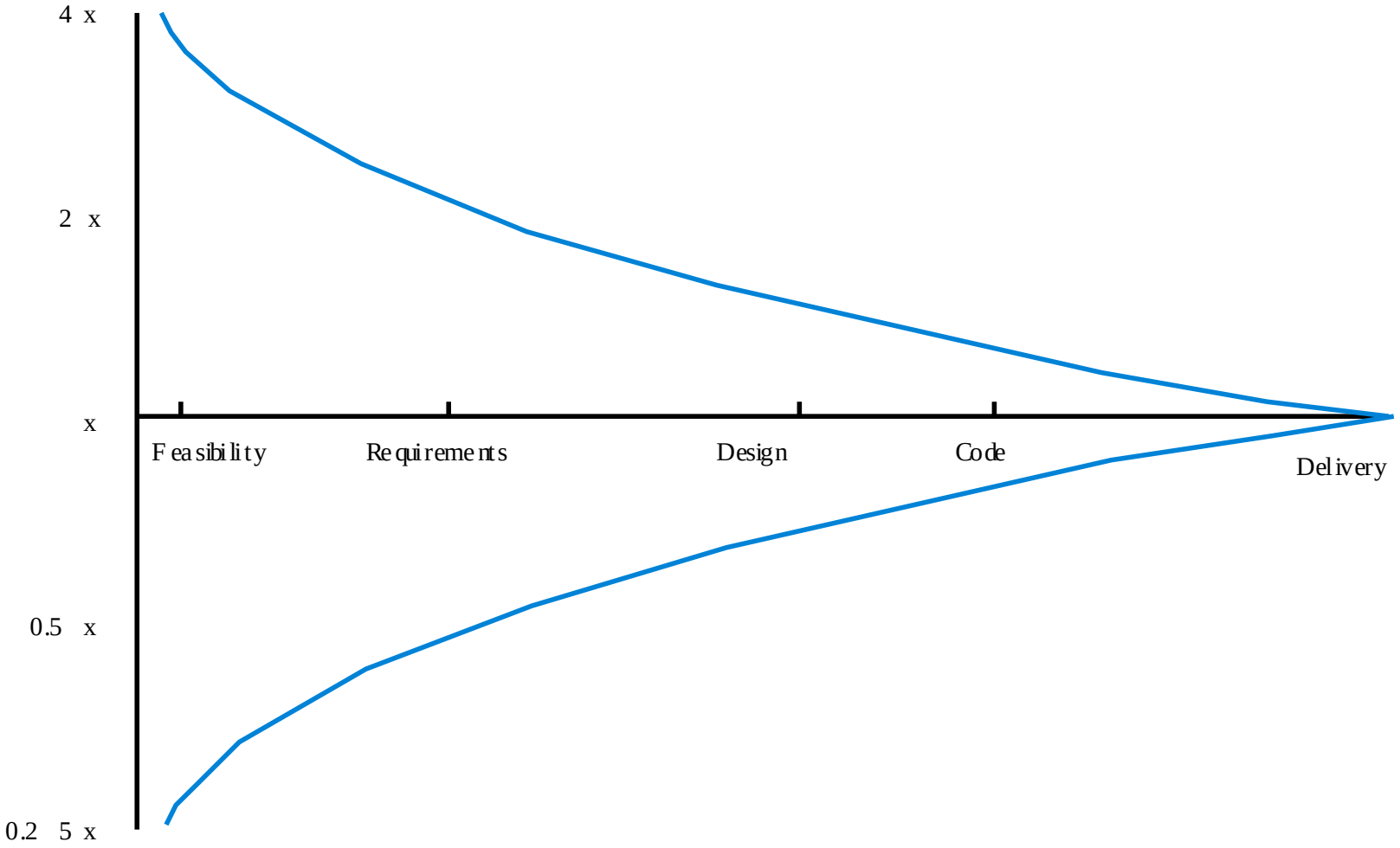
  - Veličina problema koji se rešava
    - dužina programa: LOC
  - Kvalitativni zahtevi
    - npr. visoka efikasnost, visoka bezbednost…
  - Kvalitet kreatora
    - produktivnost
  - Iskustvo kompanije sa sličnim poslovima

Vreme procene

- *Što je pre moguće*

- *na početku* faze “analize i definisanja”, na bazi preliminarne specifikacije zahteva
- *nakon* faze “analize i definisanja”, na bazi (završne) specifikacije zahteva

“Kupa” neizvesnosti procene



6. Pristupi proceni troškova

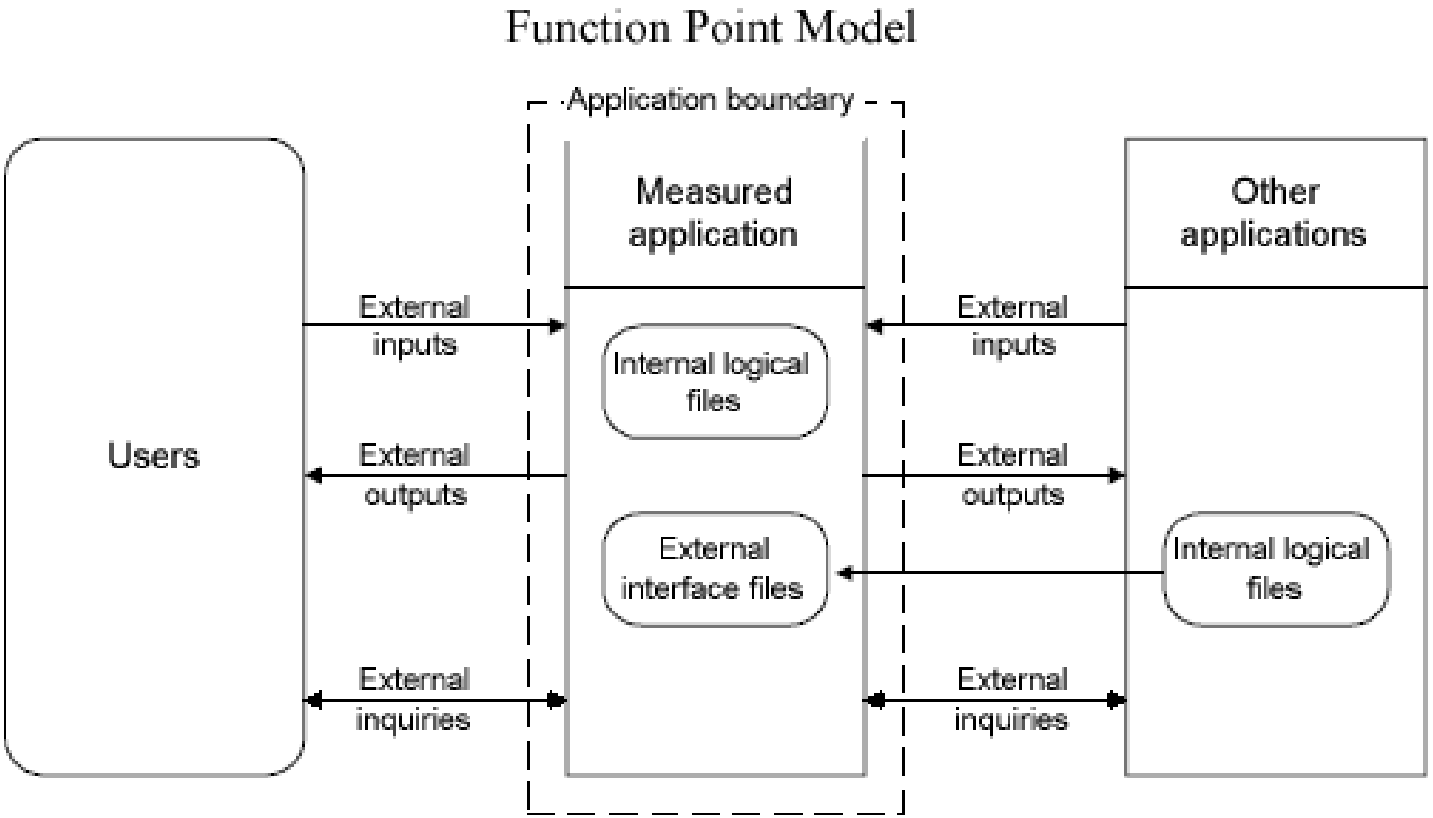
Algoritamsko modelovanje troškova	Model se zasniva na ranijoj istoriji koja povezuje neku metriku softvera (obično veličinu) sa projektnim troškovima.. Pravi se procena te metrike i model daje procenu troškova.
Ekspertska procena	Konsultuje se nekoliko eksperata za aplikativni domen i predložene razvojne tehnike.. Svaki od njih daje procenu troškova. Te se procene upoređuju i diskutuju. Proces procenjivanja iterira dok se ne postigne dogovor u proceni.
Procena po analogiji	Ova tehnika se može primeniti kada postoje raniji projekti iz istog aplikativnog domena. Trošak aktuelnog projekta se procenjuje po analogiji sa ovim ranijim projektima. Myers (1989) pruža jasan opis ovog pristupa.
Parkinsonov zakon	Parkinsonov zakon tvrdi da se rad uvek produžava da popuni raspoloživo vreme. Trošak određuju raspoloživi resursi pre nego objektivna procena. Ako se zahteva da se razvoj softvera završi za 12 meseci, a na raspolaganju je 5 programera, zahtevani resursi se mogu proceniti na 60 MM.
Pricing to win	Cena softvera je procenjena onim što mušterija ima na raspolaanju. Dakle zavisi od budžeta mušterije, a ne od funkcionalnosti softvera.

Tehnika funkcijskih poena

- Funkcijski poeni su indirektna mera količine funkcionalnosti softverske aplikacije (ne mogu se direktno meriti kao npr. LOC).
- Služe za merenje softvera kvantifikacijom funkcija procesiranja iz perspektive korisnika, povezano sa glavnim ulazima i izlazima podataka i kontrole, ili tipovima podataka.
- Prednost u odnosu na direktne mere je što se može napraviti solidna procena u ranim fazama razvoja (kada su poznati zahtevi).

Model funkcijskih poena

- Broje se spoljni ulazi (external inputs), spoljni izlazi (external outputs), spoljni upiti (external inquiries), tzv. interni logički fajlovi (ilf), eksterni interfejsni fajlovi (eif)



FP analiza za podatke (ILF,EIF)

- **Unutrašnji logički fajl (ILF)**: ILF je korisnički prepoznatljiva grupa logički povezanih podataka koja se nalazi u potpunosti u okviru granice aplikacije i održava se preko spoljnih ulaza (EI).

- **Spoljni interfejsni fajl (EIF)**: EIF je korisnički prepoznatljiva grupa logički povezanih podataka koji se koriste samo čitaju. Podaci se nalaze potpuno izvan aplikacije i održavaju od strane druge aplikacije. Eksterni interfejsni fajl je u stvari interni logički fajl za neku drugu aplikaciju.

Računanje DET, RET

- Pošto se identifikuju fajlovi potrebno im je oceniti kompleksnost. Kompleksnost podataka se određuje na osnovu dve osnovne karakteristike fajlova:
- **Tip podataka (*Data Element Type* – DET)**. predstavlja jedinstveno (bez ponavljanja) polje u ILF. Npr. ako ILF kupac ima ID to je jedan DET i njega ne treba ponovo računati kada se kupac ID pojavljuje npr. u narudžbi.
- **Tip zapisa (*Record Element Type* – RET)** grupa data elemenata unutar logičkog fajla koji se mogu ponavljati (npr. ako imamo ILF Kupac on može imati više adresa i telefona – imamo 2 RETa, Adrese i telefoni).

Pravila brojanja DET sa GUI forme

- Radio dugmad – sva povezana se broje kao jedan DET
- Check box – svaki je zaseban DET
- Komandno dugme – jedan DET za akciju koju izaziva
- Ikona, slika, zvuk – po jedan DET
- Poruke greške i potvrde – po jedan DET za operaciju na koju se odnose
- Poruka obaveštenja – broji se kao eksterni izlaz EO
- Lista za izbor – lista je EQ, rezultat izbora DET za EI

Kategorizacija ILF

- Kombinacijom broja podataka i zapisa unutrašnji fajlovi se kategorizuju faljovi kao fajlovi niske, srednje ili visoke kompleksnosti. Pravila za klasifikaciju složenosti fajlova na osnovu broja tipova podataka (DET) i tipova zapisa (RET) su prikazana u tabeli:

RET	DET		
	1-19	20-50	51 i više
1	Niska (7)	Niska (7)	Srednja (10)
2-5	Niska (7)	Srednja (10)	Velika (15)
>5	Srednja (10)	Velika (15)	Velika (15)

Kategorizacija EIF

- Kombinacijom broja tipova podataka (DET) i zapisa (RET) eksterni fajlovi se kategorizuju faljovi kao fajlovi niske, srednje ili visoke kompleksnosti. Pravila za klasifikaciju složenosti EIF:

RET	DET		
	1-19	20-50	51 i više
1	Niska (5)	Niska (5)	Srednja (7)
2-5	Niska (5)	Srednja (7)	Velika (10)
>5	Srednja (7)	Velika (10)	Velika (10)

Obrade podataka (Transakcije)

- **Spoljni ulaz (EI)**: EI je elementarni proces u kome podaci prelaze granicu od spoljašnosti aplikacije ka unutrašnjosti. Ovi podaci mogu da dolaze sa ulazne ekranske forme ili druge aplikacije. Podaci se mogu koristiti za ažuriranje jednog ili više internih logičkih fajlova (ILF). Podaci mogu biti kontrolne informacije ili poslovne informacije. Ako su podaci kontrolne informacije ne moraju da ažuriraju interni logički fajl.

Obrade podataka (Transakcije)

- **Spoljni izlaz (EO)**: EO je osnovni proces u kome sračunati podaci prolaze preko granice od unutrašnjosti ka spoljašnosti. Pored toga, EO mogu ažurirati ILF. Od podataka se kreiraju izveštaji ili izlazne datoteke poslate drugim aplikacijama. Ovi izveštaji i fajlovi su kreirani od jednog ili više ILF i EIF.

- Litmus test za identifikaciju EO je kada:

  a) podaci pređu granicu aplikacije i ažuriraju EIF,

  b) u odgovoru na poslate dobiju se neki izvedeni podaci koji ažuriraju ILF

Obrade podataka (Transakcije)

- **Spoljni upit (EQ)**: EQ je elementarni proces sa ulaznim i izlaznim komponentama koje rezultuje u dohvatanju podataka iz jednog ili više ILF i EIF, ne menjajući ILF niti EIF.
- Litmus test za identifikaciju EQ je neki izveštaj sa ulaznim kriterijumom koji dolazi od korisnika, npr. search.

Kategorizacija transakcija

- Kao i fajlovi, transakcije se klasifikuju na osnovu njihovih karakteristika na transakcije niske, srednje i visoke složenosti, na bazi vrednosti DET i FTR.
- **Tipovi podataka (*Data Element Types* – DET)**. Podaci predstavljaju nedeljive logičke informacije koje se unose u sistem ili iznose iz sistema. U zavisnosti od tipa transakcije (ulazna, izlazna ili upit) ovi podaci se mogu upisivati u fajlove sistema, prikazivati korisnicima, slati drugim sistemima i slično.
- **Referencirani tipovi fajlova (*File Type Referenced* – FTR)** svaka transakcija tokom rada koristi podatke iz fajlova sistema (ILF ili EIF)..

Kategorizacija eksternih ulaza (EI)

- Pravila za klasifikaciju eksternih izlaza:

FTR	DET		
	1-4	5-15	>15
0-1	Niska (3)	Niska (3)	Srednja (4)
2	Niska (3)	Srednja (4)	Velika (6)
>2	Srednja (4)	Velika (6)	Velika (6)

Kategorizacija eksternih izlaza (EO)

- Pravila za klasifikaciju eksternih izlaza (isto kao za EQ):

FTR	DET		
	1-5	6-19	>19
<2	Niska (4)	Niska (4)	Srednja (5)
2 ili 3	Niska (4)	Srednja (5)	Velika (7)
>3	Srednja (5)	Velika (7)	Velika (7)

Kategorizacija eksternih upita (EQ)

- Pravila za klasifikaciju eksternih izlaza (ista kao za EO):

FTR	DET		
	1-5	6-19	>19
<2	Niska (4)	Niska (4)	Srednja (5)
2 ili 3	Niska (4)	Srednja (5)	Velika (7)
>3	Srednja (5)	Velika (7)	Velika (7)

Računanje funkcijskih poena

- Ukupno računanje obavlja u tri koraka:

1. Brojanje nekorigovanih (unadjusted) funkcijskih poena, UAF
2. Računanje faktora korekcije vrednosti (value adjustment factor, VAF) na osnovu generalnih karakteristika sistema (GSC)
3. Računanje korigovanih funkcijskih poena, FP

1. Brojanje nekorigovanih funkcijskih poena

- Iz dokumentacije (modela podataka, interfejsa, specifikacije zahteva itd) broje se (ILF, EIF, EI, EO, EQ) i dodatno kategorizuju kao jednostavni, prosečni ili složeni.
  - Prema datoj tabeli svakom elementu se dodeli određeni broj funkcijskih poena
  - Zbir daje nekorigovane f.p. (UFP, unadjusted function points)

1. Brojanje nekorigovanih funkcijskih poena

Type of Component	Component Complexity			
	Low	Average	High	Total
External Input	_x3 = _	_x4 = _	_x6 = _	<== Sum
External Output	_x4 = _	_x5 = _	_x7 = _	<== Sum
External Inquiry	_x3 = _	_x4 = _	_x6 = _	<== Sum
Internal Logic File	_x7 = _	_x10 = _	_x15 = _	<== Sum
External Interface File	_x5 = _	_x7 = _	_x10 = _	<== Sum
			UAF=	*<Sum Total>*

2. Računanje faktora korekcije vrednosti VAF

- Svakom od 14 faktora iz tabele dodeli se vrednost od 0 (“ne postoji”) do 5 (“ukupan jak uticaj”)
  - Zbir tih 14 brojeva $\Rightarrow$ total degree of influence (DI), pa se VAF računa kao

    $$VAF = 0.65 + 0.01 \times DI$$

  - VAF uzima vrednosti između 0.65 and 1.35

General System Characteristic	Brief Description
1. Data Communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system? The data and control information used in the application are sent or received over communication facilities.
2. Distributed Data Processing	How are distributed data and processing functions handled?
3. Performance	Was response time or throughput required by the user?
4. Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?
5. Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?
6. On-Line data entry	What percentage of the information is entered On-Line?
7. End-user efficiency	Was the application designed for end-user efficiency?
8. On-Line update	How many ILF's are updated by On-Line transaction?
9. Complex processing	Does the application have extensive logical or mathematical processing?
10. Reusability	Is the application being developed to meet one or many user's needs?
11. Installation ease	How difficult is conversion and installation?
12. Operational ease	How effective and/or automated are start-up, back-up, and recovery procedures?
13. Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
14. Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

Funkcijski poeni (nastavak)

- 3. Finalni funkcijski poeni FP računaju se po formuli

$$FP = UAF \times VAF$$

Funkcijski poeni (nastavak)

FP se mogu koristiti za procenu veličine softvera prema postojećim podacima o prosečnom broju LOC po FPu za određeni programski jezik (AVC):

LOC = AVC * FP;

AVC varira od 200-300 za asembler do 2-40 za “spreadsheet” jezike;

Odnos između funkcijskih poena i LOC kod raznih programskih jezika

Assembler	320
C	150
COBOL	106
FORTRAN	106
Pascal	91
PL/1	80
Ada	71
Prolog	64
APL	32
C++	29*
Smalltalk	21
“Spreadsheet” jezici	6

Izvor: Yourdon, s.155; * Jones91

Parovi vrednosti funkcijskih poena (2000)

FP	IBM MM	FP	IBM MM	FP	IBM MM
50	5	700	52	1700	142
100	8	750	56	1800	153
150	11	800	60	1900	164
200	14	850	64	2000	175
250	17	900	68	2100	188
300	20	950	72	2200	201
350	24	1000	76	2300	215
400	28	1100	85	2400	230
450	32	1200	94	2500	245
500	36	1300	103	2600	263
550	40	1400	112	2700	284
600	44	1500	122	2800	307
650	48	1600	132	2900	341

Izvor: Balzert, Vol.1, 2. izdanje

FP metod: Karakteristike

- Relativno precizan
  - Eksperimenti: Devijacija od stvarne vrednosti: 10%
- Subjektivan
  - ali: u najvećoj meri nezavisan od osobe koja procenjuje
  - prosečna devijacija između 2 osobe: 12%
- Potrebno iskustvo za primenu metode
- Industrijski standard za metode procene troškova
  - širom sveta, približno 500 velikih kompanija,
    npr. Boeing, IBM, VW
- Preduslov: komercijalne aplikacije
  - npr. banke, osiguranja, uprave, stovarišta
  - ne: kompajleri i drugi sistemski softver!

FP metod: Prednosti i mane

Prednosti	Mane
Baziran na zahtevima proizvoda	Ograničen na komercijalne aplikacije
Procena moguća vremenski rano	Ima tendenciju da podceni (nekompletni zahtevi)
Jednostavan za učenje	Dobar kada se razvija novi sistem od nule, ali ne i za fazu održavanja
Dobra preciznost procene	
Podržan alatima	

Model konstruktivnih troškova (Constructive Cost Model, CoCoMo)

- Empirijski model zasnovan na iskustvima iz realnih projekata.
- Dobro dokumentovan, ‘nezavisan’ model koji nije u svojini nekog velikog proizvođača softvera.
- Dugačka istorija od inicijalne verzije objavljene 1981 (COCOMO-81) kroz različite druge verzije do COCOMO 2.
- COCOMO 2 uzima u obzir različite metodologije razvoja, višestruku upotrebu komponenata itd.

COCOMO 81

COCOMO "srednjeg nivoa"

Project complexity	Formula	Description
Simple	$MM = 2.4\ (KDSI)^{1.05} \times M$	Well-understood applications developed by small teams.
Moderate	$MM = 3.0\ (KDSI)^{1.12} \times M$	More complex projects where team members may have limited experience of related systems.
Embedded	$MM = 3.6\ (KDSI)^{1.20} \times M$	Complex projects where the software is part of a strongly coupled complex of hardware, software, regulations and operational procedures.

COCOMO (nastavak)

- M – Pomnožiti nominalnu vrednost faktorima za svaki od 15 “drajvera troška”.

Cost Drivers	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliablity	0.75	0.88	1.00	1.15	1.40	
Database size		0.94	1.00	1.08	1.16	
Product complexity	0.70	0.85	1.00	1.15	1.30	1.65
Computer attributes						
Execution time constraint			1.00	1.11	1.30	1.66
Main storage constraint			1.00	1.06	1.21	1.56
Virtual machine volatility*		0.87	1.00	1.15	1.30	
Computer turnaround time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capabilities	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Programmer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience*	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Use of modern programming practices	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

*For a given software product, the underlying virtual machine is the complex of hardware and software (operating system, database management system) it calls on to accomplish its task.