COCOMO 2

Primer izračuna časovne zahtevnosti za aplikacijo "Orodje za sledenje projekta". Aplikacija bo napisana v jeziku C# in bo uporabljala standardno relacijsko bazo podatkov.

Dekompozicija aplikacije (za razbitje na funkcijske točke):

- 9 zaslonskih mask za vnos / interakcijo z uporabnikom
- relacijska baza z identificiranimi 20 tabelami v fazi analize
- 1 poročilo z izpisom seznama vseh udeleženih po različnih projektih

Uporabili bomo COCOMO 2 model za zgodnji model načrta, za katerega velja:

- omogoča hitro in približno oceno stroškov (educated guess)
- privzamemo, da *ni dela z integracijo ponovno uporabne kode*
- primeren je za preverjanje različnih opcij implementacije uporabniških zahtev

Časovno zahtevnost projekta (v človek-mesecih) ocenimo po sledeči formuli:

$$effort_{ČM} = A * size^{B} * M$$

kjer bomo za parameter A privzeli vrednost **2.94**.

Opomba:
COCOMO II vključuje vrsto modelov, ki omogočajo vedno bolj podrobne ocene napora za izdelavo programske opreme:

1. **model sestave aplikacije**, ki se uporablja takrat, ko je programska oprema sestavljena iz obstoječih delov,
2. **zgodnji model načrta** se uporablja takrat, ko so zahteve na voljo, vendar se načrtovanje še ni začelo,
3. **model ponovne uporabe** se uporablja za izračun potrebnega dela pri integraciji komponent za večkratno uporabo,
4. **model po zasnovani arhitekturi** se uporabi, ko je zasnovana sistemska arhitektura in je na voljo več informacij o sistemu.

Ocena obsega z uporabo funkcijskih točk (določitev parametra size)

Tipi funkcionalnosti sistema

Po metodi funkcijskih točk razdelimo funkcionalnosti naše bodoče aplikacije v pet skupin:

- **EI** – Zunanji vhod (**External Input**): elementarni proces, ki sprejme in obdela podatke iz zunanjega sveta (npr. podatke, ki jih vnese uporabnik). Primer EI: obrazec za prijavo v sistem in pripadajoča krmilna logika.
- **ILF** – Notranja logična datoteka (Internal Logical File): logično zaključen skupek podatkov, ki jih vzdržujemo znotraj aplikacije. Notranja logična datoteka hrani podatke, ki jih uporablja eden ali več elementarnih procesov. Primer ILF: podatkovna baza.
- **EIF** – Zunanja vmesniška datoteka (External Interface File): logično zaključen skupek podatkov, ki jih aplikacija sicer uporablja, vzdržuje pa jih zunanji svet. EIF je vedno ILF v neki drugi aplikaciji.
- **EQ** – Zunanja poizvedba (**External Query**): elementarni proces, ki pripravi podatke za zunanji svet (npr. za uporabnika) na podlagi poizvedbe v ILF ali EIF. Primer EQ: izpis seznama registriranih uporabnikov.
- **EO** – Zunanji izhod (**External Output**): elementarni proces, ki pripravi podatke za zunanji svet (npr. za uporabnika), pri čemer pa ne upoštevamo dostopa do ILF ali EIF. V to kategorijo spadajo netrivialne obdelave podatkov. Primer EO: priprava poročila o stanju projekta.

Obseg in uteži

Ko identificiramo posamezne tipe funkcionalnosti, jim najprej določimo kvalitativni **obseg** (nizek (Low), povprečni (Average) ali visok (High)), nato pa obseg preslikamo v **utež**. Obseg in uteži določimo po sledečih pravilih:

- Obseg funkcionalnosti tipa **EI** določimo na podlagi vrednosti DET (angl. data element types) in FTR (angl. file types referenced), glej Tabelo 2.
  - Pri vnosnih obrazcih, denimo, se vrednost DET določi kot skupno **število vnosnih elementov** (vnosnih polj, potrditvenih polj, spustnih menijev...), gumbov, netrivialno izračunanih podatkov, potrditvenih sporočil, obvestil o napakah ipd. Če je neko dejanje mogoče izvesti na več načinov, oziroma če je nek podatek samo trivialna funkcija drugih podatkov, ga štejemo samo enkrat.
  - Vrednost FTR določimo kot **število ILF in EIF, do katerih moramo dostopati**, če želimo funkcionalnost realizirati.
  - Število ILF bo pri vaši funkcionalnosti enako 0, 1 ali največ 2, saj boste dostopali do največ dveh internih podatkovnih baz.
  - Število EIF pa je odvisno od tega, do koliko zunanjih podatkovnih baz dostopate v okviru funkcionalnosti.

File Types Referenced (FTR)	Data Element Types (DET)		
	1 - 4	5 - 15	16+
0 ali 1	Low	Low	Average
2	Low	Average	High
3+	Average	High	High

Tabela 1: External Input; Low = 3, Average = 4, High = 6

- Obseg funkcionalnosti tipa **EQ** ali **EO** (Tabela 3) določimo na podoben način kot obseg funkcionalnosti tipa EI. Vrednost DET se določi kot skupno **število prebranih podatkov**, netrivialno **izračunanih** podatkov, potrditvenih sporočil, obvestil o napakah ipd.

File Types Referenced (FTR)	Data Element Types (DET)		
	1 - 5	6 - 19	20+
0 ali 1	Low	Low	Average
2 ali 3	Low	Average	High
4+	Average	High	High

Tabela 2: External Query; Low = 3, Average = 4, High = 6 | External Output; Low = 4, Average = 5, High = 7

- Obseg funkcionalnosti tipa **ILF** in **EIF** določimo na podlagi vrednosti DET in RET. V primeru podatkovne baze je vrednost DET enaka **številu unikatnih stolpcev v vseh tabelah** skupaj. Če nek stolpec (npr. ID uporabnika) nastopa v več tabelah, ga štejemo samo enkrat. Vrednost RET pa določimo kot število **različnih smiselno zaokroženih skupin podatkov**, ki jih hrani podatkovna baza. Na primer, pri študijskem informacijskem sistemu bi lahko hranili podatke o uporabnikih (prva skupina), podatke o predmetih in izvajalcih (druga skupina) ter podatke o izpitih in ocenah (tretja skupina). V tem primeru bi vrednost RET torej znašala 3.

Record Element Types (RET)	Data Element Types (DET)		
	1 - 19	20 - 50	51+
1	Low	Low	Average
2 - 5	Low	Average	High
6+	Average	High	High

Tabela 3: Internal Logical File; Low = 7, Average = 10, High = 15 | External Interface File; Low = 5, Average = 7, High = 10

Ocena števila vrstic programske kode

Po pravilih, opisanih v prejšnjem razdelku, za vsako funkcionalnost določimo utež. Uteži seštejemo. Dobljeno vsoto – število funkcijskih točk – pretvorimo v ocenjeno število vrstic programske kode (angl. source lines of code, SLOC), in sicer na podlagi tabele (Function Point Languages Table), ki jo najdemo na sledeči strani QSM 2014:

https://www.qsm.com/resources/function-point-languages-table

Denimo, da želimo izdelati aplikacijo za podporo vodenju projektov po metodi Kanban. Identificirali smo sledeče funkcionalnosti:

Vrsta FP	Ime funkcionalnosti	Objekt FP	Obseg	Utež
External Input (EI)	Prijava v sistem	zaslon	LOW	3
EI	Vzdrževanje uporabnikov	zaslon	AVG	4
EI	Vzdrževanje razvojnih skupin	zaslon	LOW	3
EI	Vzdrževanje projektov	zaslon	AVG	4
EI	Kreiranje table	zaslon	AVG	4
EI	Kreiranje kartice	zaslon	LOW	3
EI	Prestavljanje kartice	zaslon	AVG	4
EI	Posodabljanje kartice	zaslon	LOW	3
EQ	Prikaz seznama uporabnikov	poročilo	LOW	3
EO	Prikaz table	poročilo	HIGH	7
Internal logical file (ILF)	Podatkovna baza	interni objekt	LOW	7
				45

Tabela 4: Izračun funkcijskih točk

Število funkcijskih točk (vsota uteži) potemtakem znaša 45. Če bomo aplikacijo razvijali v jeziku javascript, je ocenjeno število vrstic enako `45 * 47 = 2115`. Za jezik C je ocenjeno število enako `45 * 97 = 4365`. Dejansko število programskih vrstic bo lahko seveda precej odstopalo od ocenjenega. Zavedati se moramo, da je izračunana ocena zgolj izhodišče za oceno časovne zahtevnosti po metodi COCOMO II.

Razvoj bo potekal v C#. Za C# je 1FP vredna v povprečju 54 SLOC, torej:

$$size_{C\#} = \frac{FP * SLOC_{C\#}}{1000} = \frac{45 * 54}{1000} = 2.43\ KSLOC$$

V okviru razvoja se lahko uporabi različne programske jezike. Zato je potrebno tudi to upoštevati!

Parameter B

Eksponent B izračunamo naprej. B izračunamo iz 5 parametrov, ki opisujejo lastnosti projektne skupine. Parametri, ki vplivajo na eksponent, so opisani v nadaljevanju. Pri zgodnjem modelu načrta uporabljamo tipično 5 parametrov: **PREC** (ali imamo opravka s projektom nove vrste ali s poznanim), **FLEX** (kako neodvisna je skupina, je klient prisoten), **RESL** (ocenjuje pripravljenost na tveganja), **TEAM** (gre za novo ali uveljavljeno in izkušeno skupino razvijalcev) in **PMAT** (zrelostni nivo razvojnega procesa po modelu CMM). Parametre ocenimo po lestvici (Tabela 5), dodelimo točke in izračunamo eksponent B po formuli:

$$B = 1.01 + 0.01 \sum w_i$$

pri čemer so w_i uteži sledečih dejavnikov:

- **PREC**: stopnja precedenčnosti. Vrednost PREC je tem višja, čim bolj je naš novi projekt podoben projektom, na katerih smo že delali.
- **FLEX**: stopnja fleksibilnosti zahtev. Vrednost FLEX je tem višja, čim bolj so zahteve fleksibilne. Če smo se prisiljeni togo držati predpisanih zahtev, je ocena FLEX nizka.
- **RESL**: stopnja pripravljenosti na tveganja. Vrednost je tem višja, čim bolj je razvojna skupina pripravljena na morebitne ovire pri razvoju.
- **TEAM**: stopnja uigranosti skupine. Vrednost je tem višja, čim bolj je skupina medsebojno uigrana. Če je skupina sestavljena iz odličnih posameznikov, ki pa med sabo ne morejo sodelovati, je ocena TEAM nizka. Če gre za novo skupino, ki se med seboj ne pozna, je ocena prav tako nizka.
- **PMAT**: zrelostni nivo razvojnega procesa po modelu CMM. Zrelostni nivo izračunamo na podlagi odgovorov na anketo, prikazano v *tabeli I-5* v dokumentu

  https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/COCOMO/modelman.pdf

Za vsak dejavnik imamo šest možnih kvalitativnih vrednosti: ***zelo nizka, nizka, nominalna, visoka, zelo visoka, izjemno visoka*** (Tabela 5).

Zelo nizka	Nizka	Nominalna	Visoka	Zelo visoka	Izjemno visoka
5	4	3	2	1	0

Tabela 5: Kvalitativne vrednosti za dejavnike w_i. Večja vrednost pomeni manjšo utež.

Za našo skupino smo ocenili dejavnike takole:

Dejavnik	Vrednost	Utež	
PREC	Zelo nizka	5	0,05
FLEX	Nizka	4	0,04
RESL	Visoka	2	0,02
TEAM	Nizka	4	0,04
PMAT	Zelo nizka	5	0,05
		20	**0,2**

Vrednost B za naš projekt je torej: 1.01 + 0.01 * 20 = 1.01 + 0,2 = **1.21**

Parameter M

Manjka nam še izračun parametra M (angl. Effort multipliers), ki odraža sposobnost razvijalcev, nefunkcionalne zahteve, poznavanje razvojne platforme, ipd. Za *zgodnji model načrta* parameter *M* izračunamo po formuli:

$$M = \prod multiplier_i = PERS * PREX * RCPX * RUSE * PDIF * SCED * FCIL$$

pri čemer so PERS, PREX itd. uteži dejavnikov, ki jih bomo opisali v nadaljevanju. Pomen dejavnikov je sledeč:

- **PERS**: stopnja usposobljenosti članov razvojne skupine. Vrednost PERS je tem višja, čim bolj so člani usposobljeni za realizacijo zastavljenega projekta.
- **PREX**: izkušnje članov z uporabljeno tehnologijo (programskim jezikom, ogrodji, orodji, sistemom za upravljanje podatkovnih baz itd.). Vrednost PREX je tem višja, čim več izkušenj imajo člani s tehnologijo, ki jo bodo uporabili.
- **RCPX**: ocena kompleksnosti projekta; vključuje tudi zahteve glede zanesljivosti končnega izdelka. Večja kompleksnost pomeni višjo vrednost dejavnika RCPX.
- **RUSE**: potreba po izdelavi komponent, namenjenih za ponovno uporabo. Tovrstne komponente morajo biti še bistveno bolje preverjene in dokumentirane kot druge komponente. Vrednost RUSE je tem višja, čim več komponent za ponovno uporabo je treba razviti.
- **PDIF**: kombinacija spremenljivosti platforme (kako pogosto se spreminja oziroma prenavlja programski jezik ali ogrodje, ki ga nameravamo uporabiti; pogostejše spremembe pomenijo višjo vrednost PDIF) ter omejitev glede časovne in prostorske učinkovitosti aplikacije (hujše omejitve pomenijo višjo vrednost).
- **SCED**: ta parameter se uporablja za krčenje oziroma raztezanje predvidene porabe časa glede na oceno, pridobljeno v zgodnejših fazah metode COCOMO II. Za potrebe našega predmeta bo imel kar vrednost 1.0.
- **FCIL**: kombinacija razpoložljivosti razvojnih orodij (naprednejša orodja pomenijo višjo oceno) in komunikacijskih sredstev v primeru medsebojne oddaljenosti razvijalcev (naprednejša komunikacijska sredstva pomenijo višjo oceno).

Tudi tokrat imamo za vsak dejavnik šest možnih kvalitativnih vrednosti: ***zelo nizka, nizka, nominalna, visoka, zelo visoka, izjemno visoka***. Vrednosti se pri dejavnikih RCPX, RUSE in PDIF preslikajo v uteži od 0.5 (zelo nizka) do 1.5 (izjemno visoka), pri dejavnikih PERS, PREX in FCIL pa v uteži od 1.5 do 0.5 (izjemno visoka). Pri vseh dejavnikih se vrednost "nominalna" preslika v utež 1.0. (Tabela 6).

Zelo nizka	Nizka	Nominalna	Visoka	Zelo visoka	Izjemno visoka

0.5	RCPX, RUSE, PDIF	1.5
1.5	PERS, PREX, FCIL	0.5

Tabela 6: Razpon vrednosti za dejavnike

Na podlagi ocene naše skupine in projekta, smo dejavnike M ocenili takole:

Dejavnik	Opis	Ocena	Razpon uteži	Utež
PERS	Personal capability	Nominal	1.5 – 0.5	1,0
PREX	Application platform and tools experience	Low	1.5 – 0.5	1,2
RCPX	Product reliability & complexity	Very high	0.5 – 1.5	1,3
RUSE	Reusability required	Very low	0.5 – 1.5	0,5
PDIF	Platform difficulty	High	0.5 – 1.5	1,1
SCED	Schedule pressure			1,0
FCIL	Use of software tools, multisite development	Nominal	1.5 – 0.5	1,0
				0,858

Končni izračun časovne zahtevnosti

Vrednosti *A*, *size*, *B*, in *M* vstavimo v formulo za izračun:

$$effort_{PM} = 2.94 * \ 2.43^{1.21} * 0.858 = 7{,}39 \ PM$$

Ocena časovne zahtevnosti za naš projekt je torej približno **7,39** človek mesecev dela.