


## Ohjelmistotuotanto

---

### Työmäärän ja resurssien arviointi


1



### Työmäärän ja resurssien arviointi

- Projektisuunnitelmassa projektin tehtävät aikataulutetaan ja niiden suorittamiseen allokoidaan henkilöresursseja.
- Tällöin on tiedettävä paljonko resursseja työhön pitäisi allokoida ja kuinka kauan työ kestää
- Jos tiedetään työn määrä ja resurssien kyvykkyys voidaan laskea kesto


©Harri Laine 2



### Työmäärän ja resurssien arviointi

- Tee arvio niin myöhään kuin mahdollista
  - mitä enemmän on tietoa sitä helpompi on arvioida
  - jonkinlainen arvio tarvitaan kuitenkin ennen projektin käynnistystä
- Käytä hyväksi historiatietoja samankaltaisista projekteista
- Tee työn ositus suoraviivaisesti ja tavalla, jota tukevaa historia-aineistoa on saatavissa
- Käytä yhtä tai useampaa kokemusperäistä arviointimallia


©Harri Laine 3



### Työmäärän ja resurssien arviointi

- Arviointiperusteet
  - Tuoteperustainen arviointi
    - Työmäärä arvioidaan tuotteen ominaisuuksien pohjalta
  - Tuotantoperustainen arviointi
    - Työmäärä arvioidaan tuottamisprosessiin kuuluvien tehtävien pohjalta
- Useita arvioita
  - optimistinen (o), pessimistinen (p), todennäköinen(t)
  - painotettu keskiarvo esim  $(p+o+4t)/6$


©Harri Laine 4



### Tuoteperustainen arviointi

- Tuotteen ominaisuudet
  - Montako riviä ohjelmakoodia (LOC, lines of code)
    - Koodin määräkin on ennuste
    - Koodin määrä on ohjelmiston sisäinen ominaisuus
    - Eri ohjelmoijat tuottavat samasta asiasta kovin erilaisen määrän koodia
    - Helpommin johdettavissa kuin työmäärä
    - Historia-aineistot ja esimerkit auttavat
  - Toiminnallisuuden ja tietosisällön määrä ja laatu
    - Toiminnallisuuden ja tietosisällön yksiköt?
    - Perustuu ulkoisiin ominaisuuksiin
    - Analysoijat voivat päätyä merkittävästi erilaisiin mittalukuihin

©Harri Laine 5



### Koodirivien määrään perustuva arviointi

- Edellyttää toimintopohjaista ositusta hyvin pitkälle vietyinä
  - pieniä komponentteja, jotka pystytään arvioimaan
- On helppo mitata valmiista komponentista
- Yksikön keskimääräinen tuottavuus LOC/htk (rivejä/henkilötyökuukausi) on selvitettävissä historia-aineistosta
- Rivien määrä kieliriippuvaista, samoin kuin tuottavuuskin
- Soveltuu huonosti deklarativiisiin 4G kieliin

©Harri Laine 6

### Koodirivien määrään perustuva arviointi

- Erilaisia rivimääriin perustuvia mittareita
  - Löydettyjen virheiden määrä / 1000 riviä koodia (KLOC)
  - Käytössä havaittujen ohjelmistovirheiden määrä / KLOC
  - Kustannukset / LOC
  - Dokumenttisivujen määrä / KLOC
  - LOC / henkilötyökuukausi

©Harri Laine

7

### Koodirivien määrään perustuva arviointi

- Olkoon  $LOC/htk = 625$  (kun kaikki ohjelmiston tuottamiseen tehty työ mukana) ja  $Kustannukset/LOC = 50$  mk
- Olkoon ohjelmiston kokoarvio 30000 riviä
- Tällöin suoraviivaisesti:
  - Työmäärä =  $30000/625 = 48$  htk
  - Jos aikaa on käytettävissä vuosi tarvitaan 4 henkilöä
  - Kustannukset  $50 * 30000 = 1,5$  Mmk

©Harri Laine

8

### Toiminnallisuuteen perustuva arviointi

- Toimintopistemethod (Function point analysis)
  - Järjestelmän ulkoiset ominaisuudet kerryttävät toimintopisteitä
    - Syötteiden lukumäärä
    - Tulosteiden lukumäärä
    - Kyselyjen lukumäärä
    - Tiedostojen lukumäärä
    - Ulkoisten liittymien lukumäärä
  - Nämä luokitellaan yksinkertaisiksi, tavanomaisiksi tai vaikeiksi

©Harri Laine

9

### Toimintopisteiden laskenta - peruspisteet

	yksinkertaisia	*k	normaaleja	*k	vaikeita	*k	pisteet
syötteitä		*3+		*4+		*6+	
tulosteita		*4+		*5+		*7+	
kyselyjä		*3+		*4+		*6+	
tiedostoja		*7+		*10+		*15+	
liittymiä		*5+		*7+		*10	
fp:						Σ	

lukumäärät kerrotaan perässä olevalla kertoimella ja tulot lasketaan yhteen

©Harri Laine

10

### Toimintopisteiden laskenta

- Kompleksisuuskertoimilla voidaan vielä säätää hieman tulosta
- $FP = N * (0.65 + 0.01 * \Sigma(F_i))$ 
  - N= nominaalipisteet ed. kalvon taulukon mukaisena painotettuna summana
  - $F_i$  (i=1..14) kompleksisuustekijä
    - voi saada arvot 0-5
      - 0 = Ei koskaan (No influence)
      - 1 = Harvoin (Incidental)
      - 2 = Toisinaan (Moderate)
      - 3 = Keskimääräisesti (Average)
      - 4 = Merkittävästi (Significant)
      - 5 = Oleellisesti (Essential)

©Harri Laine

11

### Toimintopisteiden laskenta - kompleksisuustekijä

- Onko järjestelmä vikasietoinen? Tarvitaanko luotettavaa tietojen varmistus- ja palautusmenettelyä?
- Tarvitaanko tietoliikenneominaisuuksia?
- Onko hajautettua prosessinhallintaa?
- Onko suorituskyky kriittinen elementti?
- Käytetäänkö järjestelmää olemassaolevassa raskaassa käytössä olevassa koneympäristössä?
- Tarvitaanko interaktiivista tietojen syöttöä?
- Täytyykö interaktiivinen tietojen syöttö synkronoida usealle näytölle tai operaatiolle?
- Päivitetäänkö tiedostoja interaktiivisesti?

©Harri Laine

12

### Toimintopisteiden laskenta - kompleksisuustekijä

- Ovatko syötteen, tulosteet, tiedostot tai kyselyt monimutkaisia?
- Onko ohjelman toiminta monimutkaista?
- Onko koodi tarkoitettu uudelleenkäytettäväksi?
- Ovatko ohjelmiston muunnokset ja asennointi mukana suunnitelmassa?
- Onko ohjelmisto suunniteltu toimivaksi useina asennusvaihteluna eri organisaatioissa?
- Onko sovellus suunniteltu käyttäjätasoisiksi?

©Harri Laine

13

### Toimintopisteiden laskenta

- Toimintopistemääriä käytetään rivimäärien tapaan. Eli sillä saadaan esim. seuraavia mittareita:
  - Testauksessa löydetty virheet / FP
  - Käyttöönoton jälkeen löydetty virheet / FP
  - Kustannukset / FP
  - Dokumenttisivut / FP
  - FP / henkilötyökuukausi jne.
- Toimintopistemallista on useita muunnelmia
  - Lisätekijöitä (algoritmit), muunnettuja kertoimia

©Harri Laine

14

### Toimintopisteiden laskenta

- FP-mallin hyviä puolia:
  - Riippumaton ohjelmointikielystä
  - Perustuu dataan (ei koodiin), eli on helpommin arvioitavissa
- FP-mallin huonoja puolia:
  - Subjekttiivinen. Eri osioiden vaikeusaste riippuu tulkitsijasta
  - Mittarilla ei ole konkreettista merkitystä kuten LOC:lla (rivien lukumäärä); FP on pelkkä numero.

©Harri Laine

15

### Toimintopisteiden suhde ohjelmariveihin

- LOC:n ja FP:n välinen suhde riippuu käytetystä ohjelmointikielystä. Karkeasti voidaan johtaa seuraavanlaiset suhteet:

■ Assembler	320 LOC/FP
■ C	128 LOC/FP
■ Cobol	105 LOC/FP
■ Fortran	105 LOC/FP
■ Pascal	90 LOC/FP
■ Ada	70 LOC/FP
■ Oliokielet	30-60 LOC/FP
■ 4GL	20 LOC/FP
■ Koodigeneraattorit	15 LOC/FP
■ Taulukkolaskimet	15 LOC/FP
■ Graafiset kielet (ikonit)	4 LOC/FP

- Ylläoleva taulukko antaa arviot kielen ilmaisuvoimalle.

©Harri Laine

16

### Tuotantoperustainen arviointi

- Ositetaan työtehtävät prosessimallin mukaisesti
- esim ... määrittely, suunnittelu, toteutus
- Osuuksien suhteesta tutkimustietoa
- Edellyttää kattavaa historiatietoa
- Samanlaisuuden ongelma – onko tehtävä samankokoinen kuin projektissa X?

©Harri Laine

17

### Arviointimallit

- Edellä laskettiin hyvin suoraviivaisesti työmäärä ja henkilötarve rivimäärän ja keskituottavuuden pohjalta
- Vastaaviin laskelmiin on kehitetty tutkimuspohjaisia malleja, joilla pyritään parempiin arvioihin – mallit on kehitetty päätyneiden projektien aineistojen perusteella

©Harri Laine

18

### Arviointimallit

- Tyypillinen laskentakaava
- Työmäärä =  $A+B*(muuttuja)^C$ 
  - Muuttuja voi olla vaikka rivimäärä tai toimintopistemäärä
  - A,B ja C ovat kokeellisesti aiempia projekteja tutkimalla saatuja kertoimia

©Harri Laine 19

### Arviointimallit

- Rivimäärä pohjaisia malleja
  - $E = 5,2*(KLOC)^{0,91}$  (Waiston-Felix)
  - $E = 5,5+0,73*(KLOC)^{1,16}$  (Bailey-Basili)
  - $E = 3,2*(KLOC)^{1,05}$  (Boehm simple)
  - $E = 5,288*(KLOC)^{1,047}$  (Doty Model,  $KLOC > 9$ )
- Toimintopistepohjaisia malleja
  - $E = -13,39+0,0545*FP$  (Albrecht and Gaffney)
  - $E = 60,62*7,728*10^{-8}*FP^3$  (Kemerer)
  - $E = 585,7+15,12*FP$  (Matson, Barnett, and Mellichamp, E = työtunteja)

©Harri Laine 20

### Arviointimallit - sovellettuna

- 30000 rivin ohjelman työpanos?
  - Waiston-Felix 114 htkk => 263 LOC/htkk
  - Bailey-Basili 43 htkk => 697 LOC/htkk
  - Boehm 113 htkk
  - Doty Model 183 htkk => 163 LOC/htkk
- 625 riviä/htkk => 48 htkk

©Harri Laine 21

### Arviointimallit COCOMO-malli

- Constructive cost model
  - <http://sunset.usc.edu/research/COCOMOII/>
- Tunnetuimpia arviointimalleja (Boehm -81, -96)
- Uusin versio COCOMO II, sisältää useita erilaisia ja eri muuttujiin perustuvia kaavoja, mm. oliopisteisiin perustuvan alustavan arvioinnin kaavan. Kuitenkin perinteistä COCOMOa ovat ohjelmarivien määrään perustuvat kaavat

©Harri Laine 22

### Arviointimallit COCOMO-malli

- COCOMOn työmääräkaava
 
$$E = 2.94 * AF * KLOC^B$$
- AF= sovituskertoimen Adjustment factor, joka saadaan laskemalla mallista riippuen 7 -17 sovitustekijän tulo
- Eksponentti B määräytyy kaavalla
 
$$B = 0.91 + 0.01 * \sum_{i=1..5} S_i$$
 missä  $S_i$  :t ovat kokotekijöitä saaden kukin arvot 0-5 eli B on välillä 0.91 – 1.16  
 Kerroin 2.94 on sovituksen tulos

©Harri Laine 23

### COCOMO II: Kokotekijöitä

- Ongelman tunnettuus (täysin uusi 5 – tuttu juttu 0)
- Tavoitteiden joustavuus (tiukat tavoitteet 5, yleiset tavoitteet 0)
- Ongelmallisuus (ratkaisemattomia arkkitehtuuri ongelmia /riskejä (paljon 5, vähän 0)
- Tiimitekijät (hankala yhteistyö 5, saumaton yhteistyö 0)
- Prosessin kypsyyden CMM-tasot käänteisenä (epäkypä 5, kypsä hyvin hallittu 0)

©Harri Laine 24

### COCOMO II: Sovitustekijöitä

- Sovitustekijöitä mm.
- henkilöstön kyvykkyys, luotettavuusvaatimukset, uudelleenkäyttö, alustan vaikeus, henkilöstön kokemus, työkalujen käyttö, aikataulun tiukkuus
- Vaihteluväli 0.75 – 1.60, normaali=1
- Kaava varsin herkkä sovitustekijäkertoimien virhearvioille.

©Harri Laine

25

### COCOMO II esimerkki

- 30000 riviä sovitustekijöiltään keskimääräistä, keskinkertaisella prosessilla tutusta ongelmasta ilman suuria riskejä ja keskimääräisellä tiimillä (kukin kokotekijä olkoon 2, ja sovitustekijät keskimäärin 1)
- $E = 2.94 * 30^{1.01} = 91$  htkk

©Harri Laine

26

### COCOMO II projektin kesto

- Projektin optimikestolle COCOMOssa on kaava
- $(3.67 * E^C) * AFp/100$ , missä
  - E= työmäärä laskettuna ilman kerrointa AF
  - AFp = Sovituskertoimen AF vaikutus prosentteina (keskimääräinen=100)
  - $C = 0.28 + 0.2 * (B - 1.01)$
- eli 30000 rivin homma kestäisi
  - $3.67 * 91^{0.28} = 7,1$  kk
  - eli tarvittaisiin keskimäärin 12 henkeä ???

©Harri Laine

27

### Puttmanin ohjelmistoyhtälö

- Perustuu 4000 nykyaikaisen ohjelmistoprojektin mittauksiin. Yhtälö on:

$$E = (LOC * (B^{0.333}) / P)^3 * (1/t^4), \text{ missä}$$

- E on vaadittavat henkilötyökuukaudet
- t = projektin kesto kuukausina
- B = "Erytistaitokerroin" ("Special skills factor") joka käytännössä kasvaa sitä mukaa kun projektin koko kasvaa. Se tarkoittaa sellaisten tehtävien vaatima aikaa, jotka liittyvät testaukseen, laadunvalvontaan, ylläpitoon ja komponenttien yhteenliittämiseen). Pienille ohjelmille (KLOC=5-15) B=0.16. Isoille ohjelmille (KLOC > 70) B=0.39.

©Harri Laine

28

### Puttmanin ohjelmistoyhtälö

- P = "Tuottavuuskerroin" joka vaihtelee tehtävän ohjelmiston laadun mukaan. Tyypillisiä arvoja
  - P=2000 upotetuille tosiaikajärjestelmille,
  - P=10000 teleliikenne- ja systeemiohjelmistoille,
  - P=12000 tieteellisille ohjelmistoille ja
  - P=28000 informaatiojärjestelmä-tyyppisille ohjelmistoille

©Harri Laine

29

### Puttmanin ohjelmistoyhtälö

- Kaavasta voidaan ratkaista projektin minimikesto
- $$t_{\min} = 8.14 * (LOC/P)^{0.43} \text{ (t kuukautta, } t > 6\text{kk)},$$
- ja työmäärä
- $$E = 180 * B * t^3 \text{ (henkilötyökuukaudet, t vuotta)}$$

Eli 30000 riviä tieteellistä ohjelmistoa vaatisi

- $8.14 * (30000/12000)^{0.43} = 12$  kk
- Ja  $180 * 0.2 = 90$  htkk minimikestolla
- (huom B=0.2 on approksimaatio väliltä 0.16-0.39)

©Harri Laine

30



### Arviointimalleista

- Kaikki mallit pitää kalibroida kohteena olevaan ympäristöön. Tarvitaan historiatietoa.
- Ohjelmarivien määrä on kieliriippuva, eikä kieliriippumatonta rivimäärää voi pitää suoraan työmäärän mittana. Käsitellyissä kaavoissa 3GL-ohjelmointikielen rivimäärä (C – Pascal)