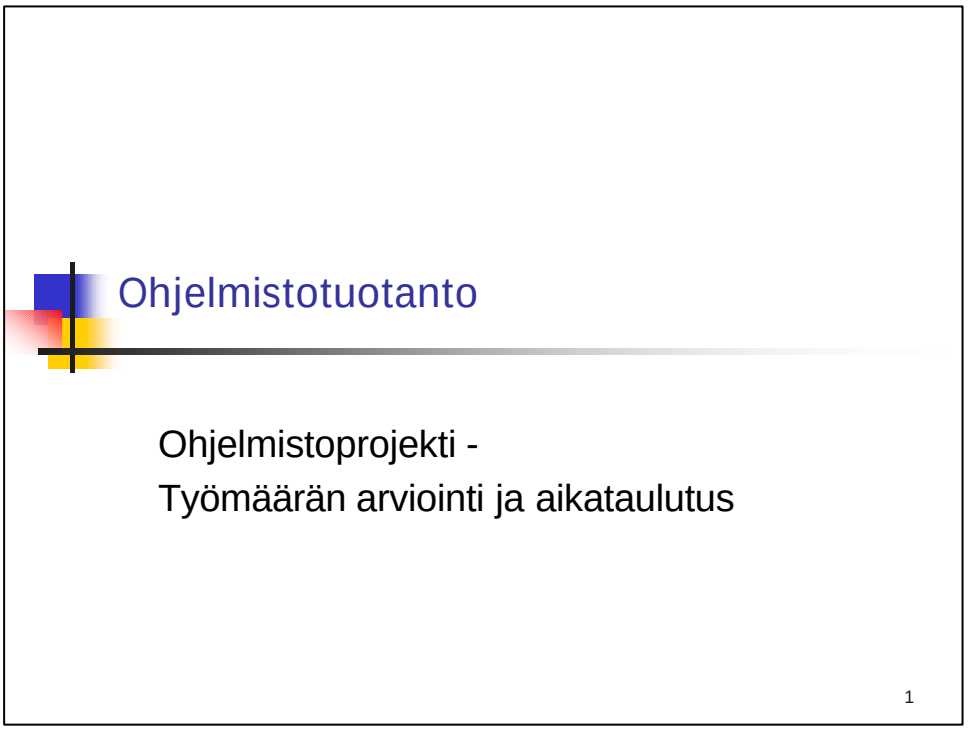
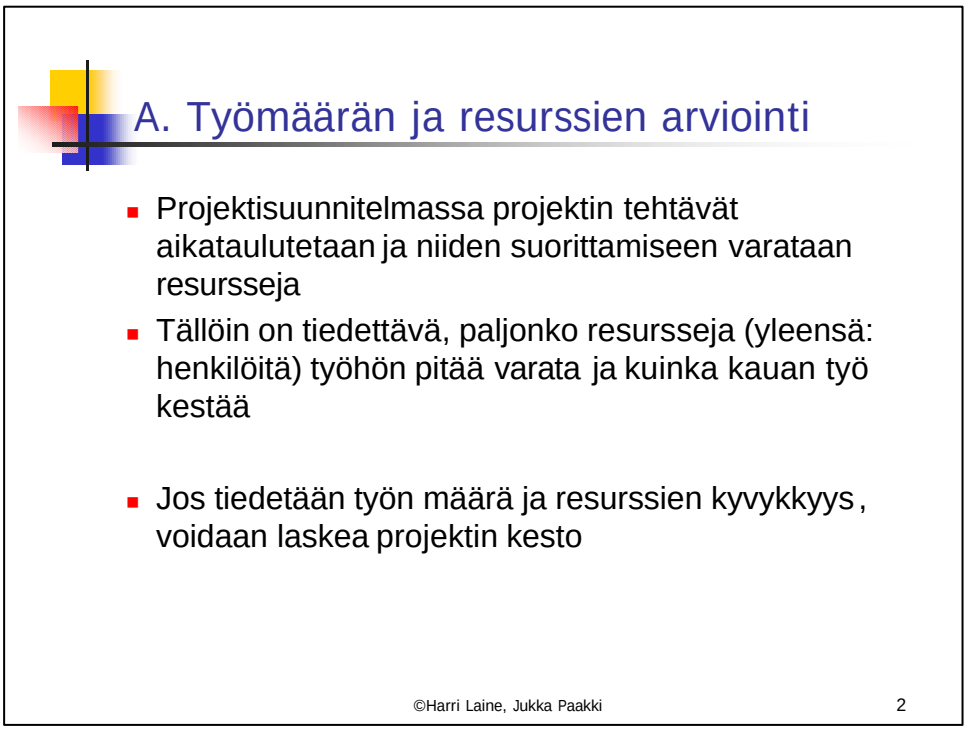

Ohjelmistotuotanto
Ohjelmistoprojekti -
Työmäärän arviointi ja aikataulutus
1


A. Työmäärän ja resurssien arviointi
Projektisuunnitelmassa projektin tehtävät aikataulutetaan ja niiden suorittamiseen varataan resursseja
Tällöin on tiedettävä, paljonko resursseja (yleensä: henkilöitä) työhön pitää varata ja kuinka kauan työ kestää
Jos tiedetään työn määrä ja resurssien kyvykkyys, voidaan laskea projektin kesto
©Harri Laine, Jukka Paakki
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 (samassa organisaatiossa)
- Tee työn ositus suoraviivaisesti ja tavalla, jota tukevaa historia-aineistoa on saatavissa
- Käytä yhtä tai useampaa kokemusperäistä arviointimallia

 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

 4

1. Tuoteperustainen arviointi

Tuotteen ominaisuudet

- Ohjelmakoodin määrä (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?**
  - **perustuvat ulkoisiin ominaisuuksiin**
  - **analysoijat voivat päätyä merkittävästi erilaisiin mittalukuihin**

 5

1.1. Koodiriviperustainen arviointi

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

 6

Koodiriviperustainen arviointi

Erilaisia rivimääriin perustuvia ohjelmistomittareita:

- löydettyjen virheiden määrä / 1000 riviä koodia (KLOC)
- käytössä havaittujen ohjelmistovirheiden määrä / KLOC
- kustannukset / (K)LOC
- dokumenttisivujen määrä / KLOC
- LOC / henkilötyökuukausi

 7

Koodiriviperustainen arviointi

- Olkoon LOC/htkk = 625 (kun kaikki ohjelmiston tuottamiseen tehty työ mukana) ja kustannukset/LOC = 10 €
- Olkoon ohjelmiston kokoarvio 30000 riviä
- Tällöin suoraviivaisesti:
  - työmäärä = 30000/625 = 48 htkk
  - jos aikaa on käytettävissä vuosi, tarvitaan optimaalisella työnjaolla 4 henkilöä
  - kustannukset = 10*30000 = 300.000 €

 8

1.2. Toiminnallisuusperustainen arviointi

- Toimintopistemenetelmä (Function Points, FP)
  - Järjestelmän ulkoiset ominaisuudet kerryttävät toimintopisteitä
    - **syötteiden lukumäärä**
    - **tulosteiden lukumäärä**
    - **kyselyjen lukumäärä**
    - **tiedostojen lukumäärä**
    - **ulkoisten liittymien lukumäärä**
  - Nämä luokitellaan yksinkertaisiksi, normaaleiksi tai vaikeiksi

 9

Toimintopisteiden laskenta - peruspisteet

	yksin-kertaisia	*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	N

lukumäärät kerrotaan perässä olevalla kertoimella ja tulot lasketaan yhteen

 10

Toimintopisteiden laskenta – kaava

- Kompleksisuuskertoimilla voidaan vielä säätää hieman tulosta

  FP = N*(0.65+0.01*$\sum(F_i)$)

- **N : nominaalipisteet ed. kalvon taulukon mukaisena painotettuna summana**
- **F_i (i=1..14) : kompleksisuustekijä, arvot 0-5**
  - **0 = Ei koskaan (No influence)**
  - **1 = Harvoin (Incidental)**
  - **2 = Toisinaan (Moderate)**
  - **3 = Keskimääräisesti (Average)**
  - **4 = Merkittävästi (Significant)**
  - **5 = Oleellisesti (Essential)**

 11

Toimintopisteiden laskenta – kompleksisuustekijät F_i

1. **Onko järjestelmä vikasietoinen? Tarvitaanko luotettavaa tietojen varmistus- ja palautusmenettelyä?**
2. **Tarvitaanko tietoliikenneominaisuuksia?**
3. **Onko hajautettua prosessinhallintaa?**
4. **Onko suorituskyky kriittinen elementti?**
5. **Käytetäänkö järjestelmää olemassaolevassa raskaassa käytössä olevassa koneympäristössä?**
6. **Tarvitaanko interaktiivista tietojen syöttöä?**
7. **Täytyykö interaktiivinen tietojen syöttö synkronoida usealle näytölle tai operaatiolle?**
8. **Päivitetäänkö tiedostoja interaktiivisesti?**

 12

Toimintopisteiden laskenta – kompleksisuustekijät F_i

9. **Ovatko syötteet, tulosteet, tiedostot tai kyselyt monimutkaisia?**
10. **Onko ohjelman toiminta monimutkaista?**
11. **Onko koodi tarkoitettu uudelleenkäytettäväksi?**
12. **Ovatko ohjelmiston muunnokset ja asennus mukana suunnitelmassa?**
13. **Onko ohjelmisto suunniteltu toimivaksi useina asennuksina eri organisaatioissa?**
14. **Onko sovellus suunniteltava käyttäjäystävälliseksi?**

 13

Toimintopisteiden käyttö

- Toimintopistemääriä käytetään rivimäärien tapaan, eli sillä saadaan esim. seuraavia mittareita:
  - testauksessa löydetyt virheet / FP
  - käyttöönoton jälkeen löydetyt virheet / FP
  - kustannukset / FP
  - dokumenttisivut / FP
  - FP / henkilötyökuukausi jne.
- Toimintopistemallista on useita muunnelmia
  - lisätekijöitä (algoritmit), muunnettuja kertoimia

 14

Toimintopisteiden käyttö

- FP-mallin hyviä puolia:
  - riippumaton ohjelmointikielestä
  - perustuu dataan (ei koodiin), eli on helpommin arvioitavissa projektin alkuvaiheissa
  - kaavoilla empiirinen (kokemusperäinen) pohja
- FP-mallin huonoja puolia:
  - subjektiivinen; eri osioiden vaikeusaste (kompleksisuustekijä) riippuu tulkitsijasta
  - mittarilla ei ole konkreettista ja intuitiivista merkitystä kuten LOC:lla (rivien lukumäärä); FP on pelkkä numero

 15

Toimintopisteiden suhde ohjelmariveihin

- LOC:n ja FP:n välinen suhde riippuu käytetystä ohjelmointikielestä. Karkeasti on voitu johtaa seuraavanlaiset suhteet:

Assembler	320 LOC/FP
C	128 LOC/FP
Cobol	105 LOC/FP
Fortran	105 LOC/FP
Pascal	90 LOC/FP
Ada95	53 LOC/FP
Oliokielet	20-60 LOC/FP (C++: 64)
Visual Basic	32 LOC/FP
SQL	12 LOC/FP
Taulukkolaskimet	15 LOC/FP
Graafiset kielet (ikonit)	4 LOC/FP

- Ylläoleva taulukko antaa arviot kielten ilmaisuvoimalle

 16

2. Tuotantoperustainen arviointi

- Ositetaan työtehtävät prosessimallin mukaisesti
- Esim: määrittely,suunnittelu, toteutus
- Osuuksien suhteesta tutkimustietoa
- Edellyttää organisaation projektit kattavaa historiatietoa
- Samanlaisuuden ongelma – onko tehtävä ”samanlainen” kuin aiemmassa projektissa X?

3. Empiiriset arviointimallit

Tyypillinen laskentakaava:

$$\text{työmäärä } E = A+B*(\text{koko})^C$$

- koko on yleensä rivimäärä (KLOC) tai toimintopistemäärä (FP)
- A,B ja C ovat kokeellisesti aiempia projekteja tutkimalla saatuja kertoimia
- työmäärä E ilmaistaan useimmiten henkilötyökuukausina (htkk)

Empiiriset arviointimallit

Rivimääräpohjaisia 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-Gaffney)
- $E = 60{,}62*7{,}728*10^{-8}*FP^3$ (Kemerer)
- $E = 585{,}7+15{,}12*FP$ (Matson-Barnett-Mellichamp, E = työtunteja)

 19

Arviointimallit - esimerkki

30000 rivin ohjelmaan 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

- Huom! Suurehko vaihtelu
  - syynä erilainen määrä erilaisia taustalla olevia empiirisiä projekteja
  - kaavan vakiot A,B ja C on säädettävä organisaation oman projektihistorian mukaisiksi

 20

3.1. COCOMO-malli

- COnstructive COst MOdel
  - http://sunset.usc.edu/research/COCOMOII/
- Tunnetuimpia arviointimalleja (Boehm 1981 ja 1996)
- Uusin versio, COCOMO II, sisältää useita erilaisia ja eri projektivaiheisiin soveltuvia malleja, mm. oliopisteisiin perustuvan alustavan (karkean) arvioinnin kaavan. Kuitenkin perinteistä COCOMOa ovat ohjelmarivien määrään perustuvat kaavat.

 21

COCOMO II

- COCOMO II:n yleinen työmääräkaava:
  $E = 2.94 * AF * KLOC^{B}$
- AF : sovituskerroin, joka saadaan laskemalla mallista riippuen 7 -17 ”sovitustekijän” (adjustment factor) tulo
- Eksponentti B määräytyy kaavalla
  $B = 0.91 + 0.01*\Sigma_{i=1..5}S_i$

  missä S_i:t ovat skaalaustekijöitä (scale factor) saaden kukin arvon 0-5, eli B on välillä 0.91 – 1.16
- Kerrointa (2.94) säädetään kulloisenkin projektitietokannan mukaan; sillä on COCOMO II:n eri versioissa ollut myös arvot 2.45 ja 2.5 (ainakin)

 22

COCOMO II: Skaalaustekijät S_i

- Projektiryhmän kokemusta ja ongelman vaikeutta kuvaavia yleisiä tekijöitä:

1. Ongelman tunnettuus (täysin uusi 5 – tuttu juttu 0)
2. Tavoitteiden joustavuus (tiukat tavoitteet 5 – yleiset tavoitteet 0)
3. Ongelmallisuus (ratkaisemattomia arkkitehtuuriongelmia / riskejä) (paljon 5 – vähän 0)
4. Tiimiysaste (hankala yhteistyö 5 – saumaton yhteistyö 0)
5. Prosessin laatu CMM-tason mukaan käänteisenä (epäkypsä prosessi 5 – kypsä hyvin hallittu prosessi 0)

 23

COCOMO II: Sovitustekijöitä AF

- Projektikohtaisia työmäärään vaikuttavia tekijöitä (sovitus- eli ”kustannustekijöitä”) mm. henkilöstön kyvykkyys, tuotteen luotettavuusvaatimukset, uudelleenkäyttöaste, aikataulun tiukkuus, henkilöstön kokemus, työkalujen käyttöaste, aikataulun tiukkuus, tuotteen mutkikkuus
- Vaihteluväli 0.70 – 1.60, normaali = 1
- Kaava varsin herkkä sovitustekijäkertoimien virhearvioille

 24

COCOMO II: Sovitustekijöitä AF

- Eniten työmäärään vaikuttavat tekijät:
  - *ohjelmistosuunnittelijoiden kyvykkyys* : maksimi / minimi = 2.24
    (eli huonoja suunnittelijoita käyttävä projekti kestää yli kaksi kertaa niin kauan kuin vastaava projekti huippusuunnittelijoilla)
  - *tuotteen mutkikkuus* : maksimi / minimi = 2.20
  - *tuotteen luotettavuusvaatimukset* : maksimi / minimi = 1.87
  - *ohjelmoijien kyvykkyys* : maksimi / minimi = 1.86

 25

COCOMO II: Esimerkki

- 30000 riviä sovitustekijöiltään keskimääräistä, keskinkertaisella prosessilla tutusta ongelmasta ilman suuria riskejä ja keskimääräisellä tiimillä (kukin skaalaustekijä kaavan eksponentissa B olkoon 2 ja sovitustekijät kertoimessa AF keskimäärin 1)
- $E = 2.94 * 1 * 30^{1.01} = 91$ htkk
  (eli reilut 8 henkilötyövuotta)

 26

COCOMO II: Projektin kesto

- Projektin optimikestolle (kk) on COCOMOssa kaava $(3.67 * E^{C}) * AFs/100$, missä
  - E = työmäärä laskettuna ilman sovitustekijää ”aikataulun tiukkuus”
  - AFs = sovitustekijän ”aikataulun tiukkuus” vaikutus prosentteina (löysä aikataulu=75, realistinen/COCOMO=100, tiukka aikataulu=160)
  - C = 0.28 + 0.2*(B-1.01)
- Esimerkiksi edellinen 30000 rivin homma:
  - $3.67 * 91^{0.28} = 7,1$ kk
  - eli tarvittaisiin keskimäärin 13 henkeä ???

 27

3.2. Puttmanin ohjelmistoyhtälö

- Perustuu 4000 nykyaikaisen ohjelmistoprojektin mittauksiin:

  $E = (LOC*(B^{0,333})/P)^3*(1/t^4)$

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

 28

Puttmanin ohjelmistoyhtälö

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

 29

Puttmanin ohjelmistoyhtälö

- Kaavasta voidaan ratkaista projektin minimikesto

  $t_{min} = 8.14*(LOC/P)^{0.43}$ (t kuukautta, t > 6kk),
- ja työmäärä

  $E = 180*B*t^3$ (henkilötyökuukaudet, t vuotta)
- Esim. 30000 riviä tieteellistä ohjelmistoa:
  - $8.14*(30000/12000)^{0.43}$ = 12 kk
  - 180*0.2 = 90 htkk minimikestolla
  - (huom B=0.2 on approksimaatio väliltä 0.16-0.39)

 30

Empiirisistä arviointimalleista

- Kaikki mallit (vakiotekijät) pitää säätää kohteena olevaan ympäristöön. Tarvitaan historiatietoa.
- Ohjelmarivien määrä on kieliriippuva, mutta se voidaan johtaa esimerkiksi kieliriippumattomasta FP-arvosta. Kaavoissa oletetaan yleensä, että käytetään ns. 3. sukupolven ohjelmointikieltä (C-Pascal-Ada).
- (Vanhemmat) mallit olettavat, että käytetään vesiputousprosessia, jolloin ne eivät päde sellaisinaan nykyaikaisiin prosesseihin (XP!?)

 31

B. Projektiaikataulu

- Aikataulun suunnittelun lähtökohtana ovat
  - laadittu työnositus ja
  - tehtävien resurssitarpeet (työmäärä, muut)
- Työnosituksen yhteydessä on syytä selvittää tehtävien väliset riippuvuudet
  - mikä tehtävä on ennen jotain toista tehtävää, esim. suunnittelu ennen toteutusta
- Tehtävän kesto riippuu sille varattavista resursseista ja työmäärästä
  - resurssien tuplaaminen ei kuitenkaan välttämättä puolita kestoa (Brooks: The Mythical Man-Month)

 32

Projektiaikataulu - tehtävien riippuvuudet

T2 ja T4 suoritettava loppuun ennen T5:n aloittamista

Tehtävä	Kesto (päiviä)	Edeltäjät
T1	8	
T2	15	
T3	15	T1
T4	10	
T5	10	T2,T4
T6	5	T1,T2
T7	20	T1
T8	25	T4
T9	15	T3,T6
T10	15	T5,T7
T11	7	T9
T12	10	T11

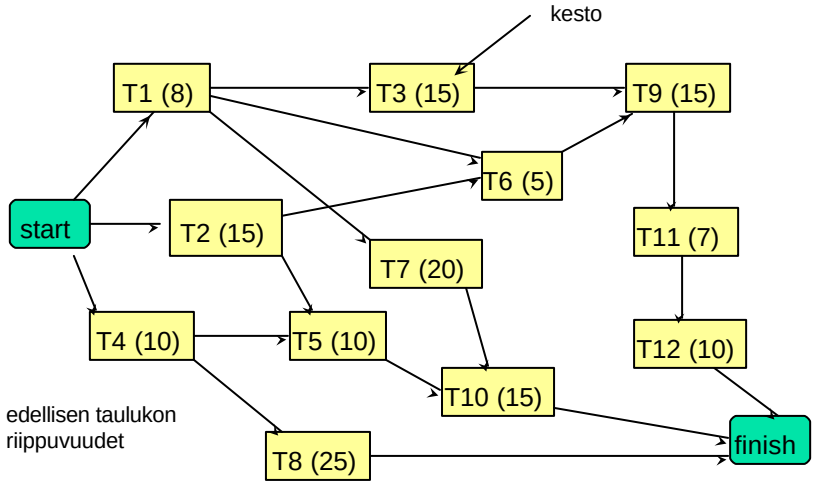
 33

Projektiaikataulu

- Tehtäväverkko (activity/task network)
  - esitetään tehtävien riippuvuudet graafisesti
  - voidaan määritellä etappeja (milestone) eli kohtia, joissa tarkistetaan, onko projekti aikataulussaan
- Kriittinen polku: tehtäväketju, jossa jonkin tehtävän myöhästyminen samalla myöhästyttää koko projektia
  - kriittisiä polkuja voi olla useita
  - määrättävissä tehtäväverkon perusteella
- Tehtäväverkon aikataulutus = tehtävien sitominen kalenteriin
  - jätettävä pelivaraa yllättäviä myöhästymisiä varten
  - myös projektin hallintaan jätettävä pelivaraa
  - kriittisen polun ulkopuolella on aina pelivaraa

 34

Tehtäväverkko (tehtävät/kestot/riippuvuudet)



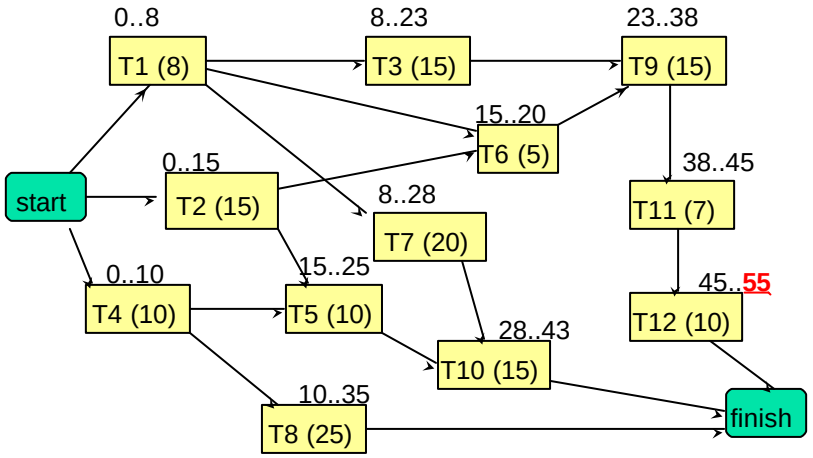
 35

Tehtäväverkko (tehtävät/kestot/riippuvuudet)

- Tehtäväverkon avulla voidaan helposti laskea:
  - aikaisin alkamisaika (= aika, jolloin kaikki aikaisemmat osavaiheet ovat päättyneet minimiajassa)
  - aikaisin päättymisaika (aikaisin alkamisaika + tehtävän kestoaika)
  - viimeisin alkamisaika (aika, jolloin tehtävän on viimeistään alettava, jotta projekti saadaan ajallaan päätökseen)
  - viimeisin päättymisaika (viimeisin alkamisaika + tehtävän kestoaika)

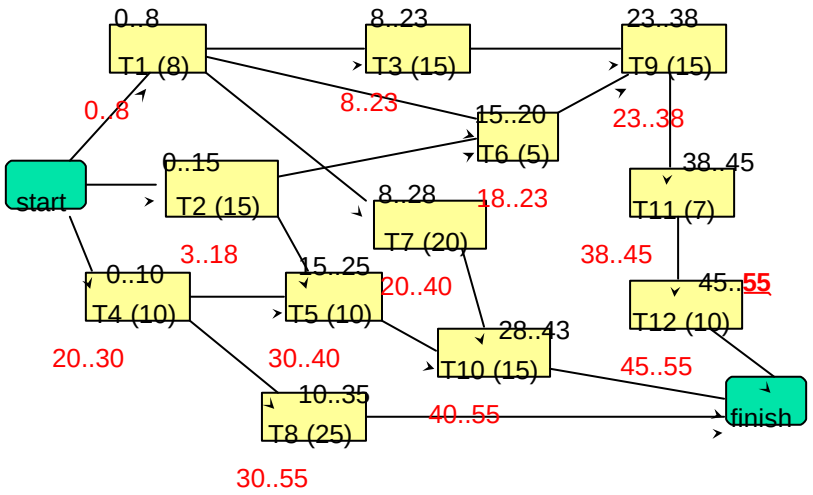
 36

Aikaisin alkamisaika ja päättymisaika



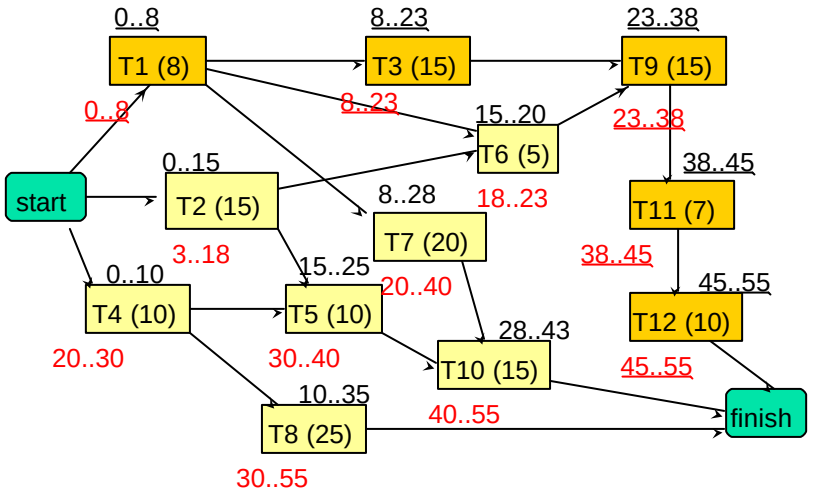
 37

Viimeisin alkamisaika ja päättymisaika



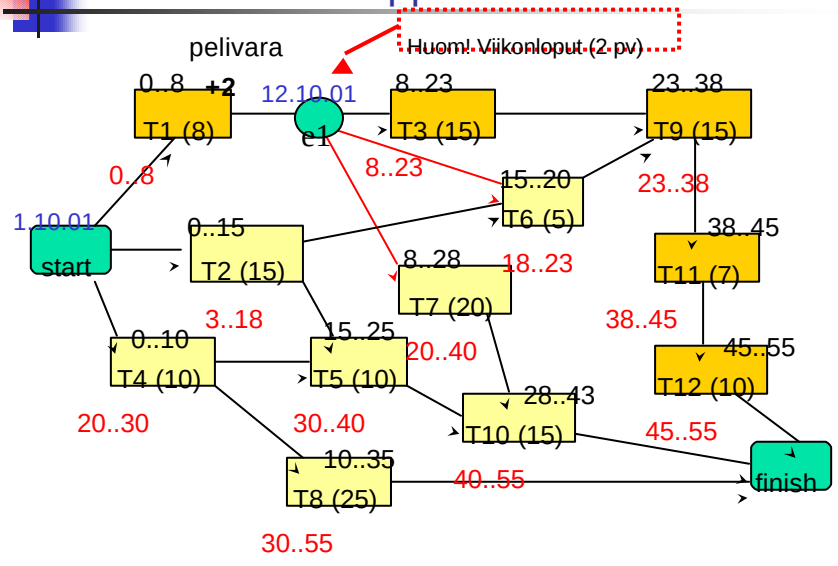
 38

Kriittinen polku



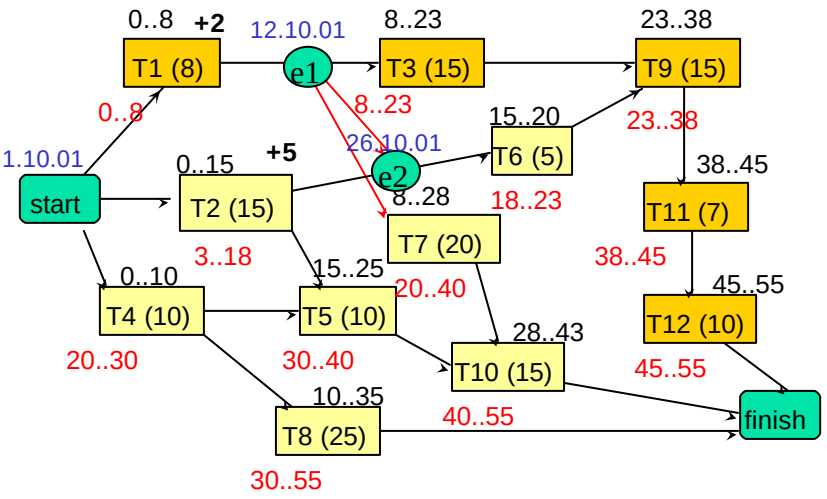
 39

Aikataulutus etappien avulla



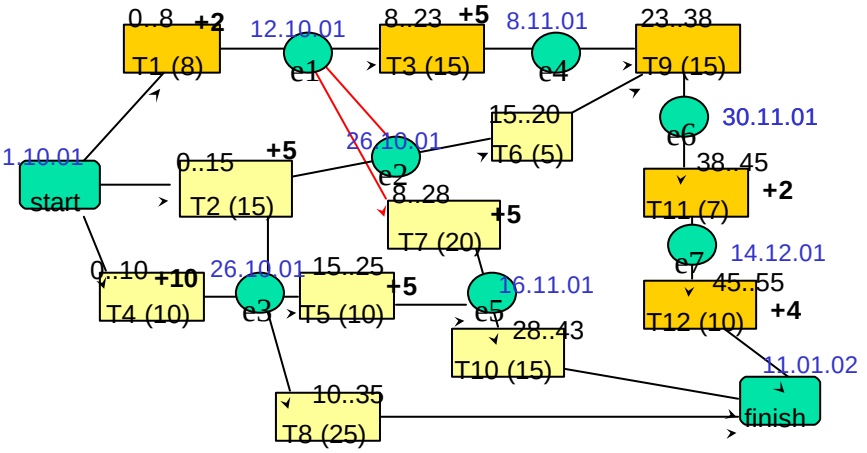
 40

Aikataulutus etappien avulla



 41

Aikataulutus etappien avulla



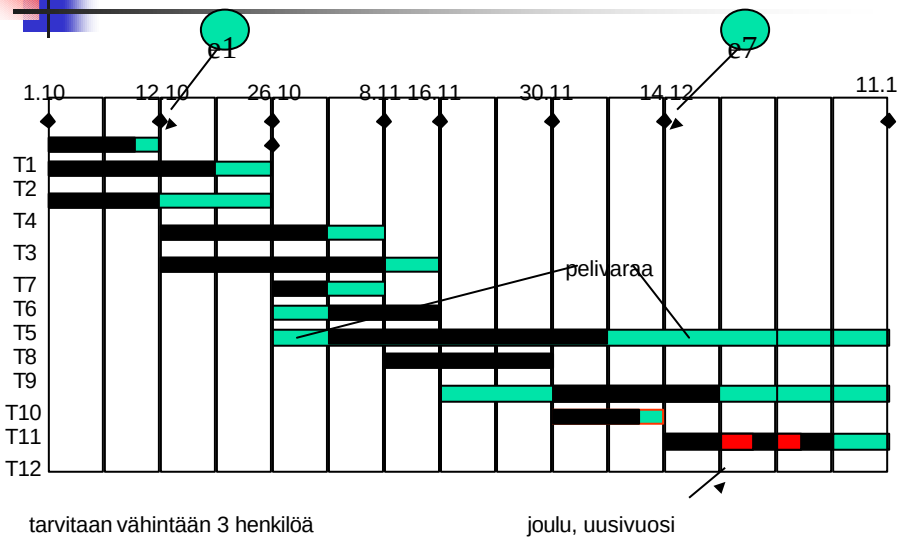
 42

Aikataulutus etappien avulla

- Esimerkkiprojektin kokonaiskesto siis 15 viikkoa = 105 päivää, joista työpäiviä 68
  - kriittiselle polullekin on siis jätetty pelivaraa 13 työpäivää
  - aikataulu edellyttää projektin resurssointia siten, että tehtäviä voidaan tehdä rinnakkain
  - GANTT-kaavio (GANTT chart) (janakaavio) esittää projektin aikataulun kalenterissa

 43

Gantt-kaavio



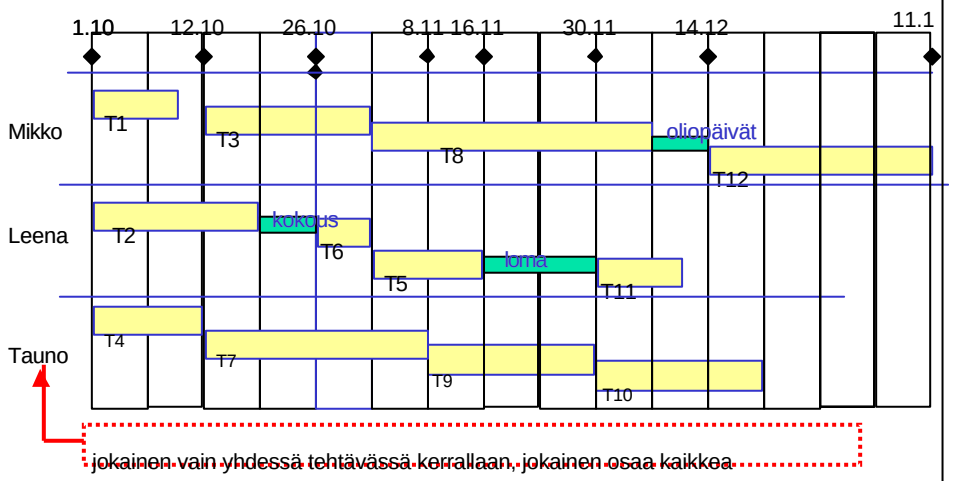
 44

Projektin aikataulu

- Aikataulutusta laadittaessa otettava huomioon (henkilö)resurssit
  - montako tehtävää voidaan tehdä rinnakkain
    - henkilöstön henkilökohtaiset kalenterit
      - kokoukset, matkat, lomat
  - kuka osaa tehdä mitäkin tehtävää
  - useamman projektin jakamat resurssit

 45

Projektin aikataulu



 46