


## Ohjelmistotuotanto

---

### Ohjelmistojen ylläpito


1



## Ohjelmistojen ylläpito

- Ohjelmistojen ylläpito (software maintenance) on ohjelmiston elinkaaren vaiheista pitkäkestoisin ja kallein: jopa yli 70% pitkäikäisen ohjelmiston kokonaiskustannuksista
- Ylläpito: jostakin tietystä syystä ohjelmistoon kohdistettava muutostyö
  - katsotaan useimmiten alkavaksi sen jälkeen kun ohjelmito on luovutettu asiakkaalle tai viety markkinoille
  - suuri kustannusvaikutus johtuu muutoksista elinkaaren viimeisessä vaiheessa


© Harri Laine, Jukka Paakki 2



## Ohjelmistojen ylläpito

- Ylläpidon lajit:
  - korjaava (corrective): tuotanto/testausvaiheessa paljastumatta jääneiden virheiden korjaus
    - osuus 21% ylläpidosta
  - mukauttava (adaptive): ohjelmiston siirtäminen uuteen käyttöympäristöön
    - osuus 25%
  - täydentävä (perfective): käyttäjien uusien tarpeiden tai vaatimusten toteuttaminen
    - osuus 50%
  - ehkäisevä (preventive): ohjelmiston parantaminen tulevia ylläpitotarpeita varten
    - osuus 4%


© Harri Laine, Jukka Paakki 3



## Ohjelmistojen ylläpito

- Ylläpidon yleiset osatehtävät:
  1. Ongelmien, virheiden tai vaatimusten tunnistaminen.
  2. Ohjelmistoon (koodiin, dokumentaatioon) perehtyminen.
  3. Tarvittavien muutosten suunnittelu.
  4. Muutettavien ohjelmito-osien etsiminen.
  5. Osien toiminnan ymmärtäminen.
  6. Ohjelmito-osien muuttaminen.
  7. Muutosten vaikutusanalyysi: heijastuvatko muutokset muihin ohjelmiston osiin ja onko niitäkin muutettava?
  8. Muutosten (regressio)testaus.


© Harri Laine, Jukka Paakki 4



## Ohjelmistojen ylläpito

- Ylläpitäjän tietotarpeita ja -lähteitä:
  - Ohjelmiston tarkoitus: dokumentaatio.
  - Sovellusalueen käsitteet ja menetelmät: kokemus, oppikirjat, asiantuntijat.
  - Ohjelmiston toiminta: dokumentaatio, koodi, suoritusdata, ohjelmiston kehittäjät, visualisointityökalut.
  - Muutosten turvalliset toteuttamistavat: oppikirjat, asiantuntijat, työkalut (koodigeneraattorit, refaktorointit).
  - Ohjelmiston kehityshistoria: pöytäkirjat, dokumentaatio, versionhallinta.

© Harri Laine, Jukka Paakki 5



## Ohjelmistojen ylläpito – takaisinmallinnus

- Ohjelmiston takaisinmallinnus (reverse engineering): ohjelmiston osien ja niiden välisten suhteiden tunnistaminen ja esittäminen korkeammalla abstraktiotasolla
  - koodin osalta suhteellisen hyvin automatisoituva alue
  - koodi → arkkitehtuuri (hierarkkinen rakenne, luokkakaavio)
  - koodi → suunnitteluratkaisut (suunnittelumallit)
  - koodi → kontrollin kulku (kutsukaavio, vuokaavio)
  - koodi → datan kulku (viipaleet)
  - koodi → osien tai olioiden yhteistyö (sekvenssikaavio)
  - visualisointi, animointi

© Harri Laine, Jukka Paakki 6

### Takaisinmallinnus – HyperSoft

- HyperSoft: väline ohjelmistojen takaisinmallinnukseen, ymmärtämiseen ja ylläpitoon
  - kehitetty Jyväskylän yliopistossa ja Helsingin yliopistossa
  - C, upotettu SQL
  - staatinen analyysi
  - koodi → hypertexti
  - graafiset, abstraktit, osittaiset näkymät
  - moduulien (C-tiedostojen) rajat ylittävä linkitys
  - hypertekstiseläimen mukainen käyttöliittymä

© Harri Laine, Jukka Paakki 7

### Takaisinmallinnus – HyperSoft

- HyperSoftin hakurakenteet:
  - määrittelyviite (definition reference): linkki tunnuksesta sen määrittelyyn (vakiot, muuttujat, tyypit, funktiot)
  - esiintymälista (occurrence list): kaikki valitun tunnuksen esiintymät ohjelmassa (vakiot, muuttujat, tyypit, funktiot)
  - ilmentymälista (instance list): kaikki tietyn syntaktisen tyyppin ilmentymät, esim. kaikki sijoitus- tai tulostuslauseet
  - kutsukaavio (call graph): funktioiden tai moduulien väliset kutsuketjut, eteen- tai taaksepäin suhteessa kutsusuuntaan
  - viipale (slice): tiedonkulku (data flow)
    - taaksepäin viipalointi (backward slicing): ohjelman osat, jotka saattavat vaikuttaa valitun muuttujaesiintymän arvoon
    - eteenpäin viipalointi (forward slicing): ohjelman osat, joihin valitun muuttujaesiintymän arvo saattaa vaikuttaa

© Harri Laine, Jukka Paakki 8

### Takaisinmallinnus – HyperSoft

Muuttujan esiintymälistä

```

read(i);
if ((i < 0) || (i > 100))
    error();
else
    {sum=0; x=0;

    while (x < i)

        { x=x+1;

         if (i==10) sum=1; else sum=sum+x; }

    print (sum); }
    
```

© Harri Laine, Jukka Paakki 9

### Takaisinmallinnus – HyperSoft

Sijoituslauseiden esiintymälistä

```

read(i);
if ((i < 0) || (i > 100))
    error();
else
    {sum=0; x=0;

    while (x < i)

        { x=x+1;

         if (i==10) sum=1; else sum=sum+x; }

    print (sum); }
    
```

© Harri Laine, Jukka Paakki 10

### Takaisinmallinnus – HyperSoft

Taaksepäin viipalointi, esimerkiksi

```

read(i);
if ((i < 0) || (i > 100))
    error();
else
    {sum=0; x=0;

    while (x < i)

        { x=x+1;

         if (i==10) sum=1; else sum=sum+x; }

    print (sum); }
    
```

© Harri Laine, Jukka Paakki 11

### Takaisinmallinnus – HyperSoft

Eteenpäin viipalointi, esimerkiksi

```

read(i);
if ((i < 0) || (i > 100))
    error();
else
    {sum=0; x=0;


    while (x < i)

        { x=x+1;

         if (i==10) sum=1; else sum=sum+x; }

    print (sum); }
    
```

© Harri Laine, Jukka Paakki 12



Ohjelmistotuotanto

---

Ohjelmistojen mittaaminen

13

Ohjelmistojen mittaaminen

- Ohjelmistomittarit (software metrics): ohjelmiston tai ohjelmistoprojektin ominaisuuksia kuvaavia kvantitatiivisia (numeerisia) arvoja
- Perinteisiä mittareita:
  - toimintopisteet (function points): ohjelmiston koko
  - COCOMO: ohjelmistoprojektin työ määrä
  - moduulien kytkentä (coupling) ja kiinteytys (cohesion)
  - "ohjelmistotiede" (software science, Halstead 1977): ohjelmiston koko ja sen toteutuksen vaativuus
  - "syklomaattinen kompleksisuusmitta" (McCabe, 1976): ohjelman mutkikkuus

© Harri Laine, Jukka Paakki 14

Ohjelmistojen mittaaminen

- Ohjelmistomittarien ja laadun suhde: tavoitteena arvioida ohjelmiston tai sen tuotantoprosessin laatua kvantitatiivisten mittarien avulla
  - toimintopisteet ⇒ ohjelmistokoodin määrä
  - syklomaattinen kompleksisuus ⇒ testaamisen ja ylläpidon vaatima työ määrä
- Suhde epäselvä
  - esim. korrelaatio *syklomaattinen kompleksisuus*  $\mathcal{D}$  *ylläpidon työ määrä* näyttää olevan pikemminkin seuraus siitä, että mutkikkaat ohjelmat ovat suuria ja siksi vaikeita ylläpitää
  - *ohjelmiston koko*  $\mathcal{D}$  *työ määrä* varmin ja useimmiten taustalla oleva peruskorrelaatio

© Harri Laine, Jukka Paakki 15

Olioperustaiset mittarit

- S.R. Chidamber, C.F. Kemerer: A Metrics Suite for Object-Oriented Design. *IEEE Transactions on Software Engineering*, 20, 6, 1994, pp. 476-493.
  - kuusi olioperustaista mittaria
  - vakiintunut asema
  - pohja myöhemmille mittareille
  - soveltuvat sekä suunnitelman että koodin mittaamiseen

© Harri Laine, Jukka Paakki 16

Olioperustaiset mittarit

- Weighted methods per class (WMC): luokan kokonaismutkikkuus  $WMC(C) = \sum c_i$  missä  $c_i$  on luokan  $C$   $i$ :n metodin mutkikkuus (esim. McCabe)
- Depth of inheritance tree (DIT): luokkahierarkian syvyys  $DIT$  = hierarkian juurisolmusta lehtisolmuun johtavan pisimmän polun pituus
- Number of children (NOC): luokan välittömien jälkeläisten määrä  $NOC(C)$  = luokan  $C$  aliluokkien lukumäärä
- Coupling between object classes (CBO): luokan kytkentä  $CBO(C)$  = luokan  $C$  käyttämien muiden luokkien lukumäärä (käyttö: perintä, koostuminen, kutsu)

© Harri Laine, Jukka Paakki 17

Olioperustaiset mittarit

- Response for a class (RFC): metodikutsun ulottuvaisuus  $RFC(m) = |M \cup R_i|$  missä  $m$  on luokasta  $C$  kutsuttu metodi,  $M$  on luokan  $C$  metodien joukko ja  $R_i$  on  $C$ :n  $i$ :n metodin kutsumien metodien joukko
- Lack of cohesion in methods (LCOM): kiinteyden puute  $LCOM(C)$  = niiden luokan  $C$  metodien lukumäärä, jotka käyttävät yhteisiä  $C$ :n attribuutteja tai ilmentymämuuttujia; maksimaalista kiinteyden puutetta edustaa arvo  $LCOM = 0$ , kun taas hyvää kiinteyden astetta edustavat arvot  $LCOM > 0$

© Harri Laine, Jukka Paakki 18

### Olioperustaiset mittarit

```

classDiagram
    class C
    class C1
    class C2
    class C11
    class C21
    class C22
    class C23
    class C211
    C <|-- C1
    C <|-- C2
    C1 <|-- C11
    C2 <|-- C21
    C2 <|-- C22
    C2 <|-- C23
    C21 <|-- C211
    C1 --> C2
    
```

*DIT* = 4  
*NOC* (C2) = 3  
*NOC* (C11) = 0  
*CBO* (C1) = 3  
*CBO* (C) = 0

© Harri Laine, Jukka Paakki 19

### Olioperustaiset mittarit

- Weighted methods per class (WMC): luokan toteuttamiseen ja testaamiseen tarvittava työmäärä
- Depth of inheritance tree (DIT): suunnitelman / koodin ymmärtämiseen ja testaamiseen tarvittava työmäärä, suunnitelman / koodin uudelleenkäyttöaste
- Number of children (NOC): luokan testaamiseen tarvittava työmäärä, luokan uudelleenkäyttöaste
- Coupling between object classes (CBO): luokan muuttamiseen tarvittava työmäärä, luokan uudelleenkäytettävyys käänteisenä
- Response for a class (RFC): luokan testaamiseen tarvittava työmäärä
- Lack of cohesion in methods (LCOM): luokan mutkikkuus, luokan muuttamiseen tarvittava työmäärä

© Harri Laine, Jukka Paakki 20