

JUSSI KOSKINEN

OHJELMISTOJEN YLLÄPITO JA EVOLUUTIO

Moniste:

- Koskinen, J., Sakkinen, M. & Paakki, J. ”Ohjelmistotekniikka” (2 p.). *Tietojenkäsittelytieteen julkaisuja, Opetusmonisteita OM-10*. Dept. of Computer Science and Information Systems, Univ. of Jyväskylä, Jyväskylä, Finland. Jyväskylän Yliopistopaino. Dec. 2001. 234 p. Table of contents: <http://www.cs.jyu.fi/~koskinen/ohtelms.htm> (s. 118-152).

Kurssit:

- ”Ohjelmistojen ylläpito” (ITK S63)
<http://www.cs.jyu.fi/~koskinen/oyp0.htm>
- ”Ohjelmistojen ylläpidon tehtävät” (ITK S61)
<http://www.cs.jyu.fi/~koskinen/oyp1.htm>
- ”Ohjelmistojen ylläpidon tekniikat” (ITK S62)
<http://www.cs.jyu.fi/~koskinen/oyp2.htm>
- ”Ohjelmistojen ylläpidon kustannusten arviointi” (ITK S64)
<http://www.cs.jyu.fi/~koskinen/oyp3.htm>

Seminaarit:

- [Ohjelmistojen ylläpidon seminaari – 2003/Loppuraportin alkuosa](http://www.cs.jyu.fi/~koskinen/ohteyps03.htm)
<http://www.cs.jyu.fi/~koskinen/ohteyps03.htm>
- [Ohjelmistojen ylläpidon seminaari – 2002/Loppuraportin alkuosa](http://www.cs.jyu.fi/~koskinen/ohteyps02.htm)
<http://www.cs.jyu.fi/~koskinen/ohteyps02.htm>
- [Ohjelmistojen ylläpidon seminaari – 2001/Loppuraportin alkuosa](http://www.cs.jyu.fi/~koskinen/ohteyps01.htm)
<http://www.cs.jyu.fi/~koskinen/ohteyps01.htm>

Bibliografiat ym.:

- [Ohjelmistojen ylläpidon kirjojen luettelo](http://www.cs.jyu.fi/~koskinen/bibsmb.htm)
<http://www.cs.jyu.fi/~koskinen/bibsmb.htm>
- Ohjelmistojen ylläpidon merkitys,
<http://www.cs.jyu.fi/~koskinen/smcosts.htm>
- [Ylläpidon kustannusten arviointi ja ohjelmistojen modernisointi](http://www.cs.jyu.fi/~koskinen/bibelt.htm) (ELTIS-projekti) <http://www.cs.jyu.fi/~koskinen/bibelt.htm>
- [Käänteistekniikoiden bibliografia](http://www.cs.jyu.fi/~koskinen/bibre.htm) (HyperSoft-projekti)
<http://www.cs.jyu.fi/~koskinen/bibre.htm>

OHJELMISTOJEN YLLÄPITO

YLLÄPIDON YLEINEN LUONNE:

- ◇ **Ylläpito: ohjelmiston kehitystä sen käyttöönoton jälkeen**
- ◇ **Ylläpito on tärkeä osa ohjelmistojen elinkaarta (jopa n. 70 % kustannuksista)**
- ◇ **Ylläpidossa keskeistä on ohjelmiston muuttaminen**
- ◇ **Ohjelmiston muuttaminen voi edellyttää paluuta aikaisempiin vaiheisiin**
- ◇ **Ohjelmiston tuottaminen vs muuttaminen**
- ◇ **Ylläpito vs uudelleenkäyttö**

SYSTEMAATTINEN YLLÄPITO:

- ◇ **Kehitteillä olevien ohjelmistojen ylläpidettävyys:**
laatu, ymmärrettävyys, joustavuus, ...
- ◇ **Olemassaolevien ohjelmistojen ylläpidettävyys:**
dokumentointi, uudelleenmuokkaus, ...
- ◇ **Olemassaolevien ohjelmistojen ylläpidon tuki:**
prosessitietämys, konfiguraationhallinta,
apuvälineet,...

MUUTOKSET:

- ◇ **Lähdekoodin muutokset**
- ◇ **Dokumenttaation muutokset**
- ◇ **Muutokset ennen/jälkeen ohjelmiston käyttöönoton**

YLLÄPIDON LAJIT (LIENTZ & SWANSON, SOMMERVILLE)

- 1) Korjaava ylläpito: virheet (17 %)**
- 2) Mukauttava ylläpito: tekniset muutokset (18 %)**
- 3) Täydentävä ylläpito: käyttäjätarpeet (65 %)**
- 4) Ehkäisevä ylläpito: ylläpidettävyys**

YLLÄPIDON TUTKIMUSKOHTEET (HAWORTH YM.):

- ◇ **Ohjelmoija/ylläpitäjä: koulutus, kokemus, taidot, ...**
- ◇ **Ohjelmakoodi: käytetyt ohjelmointikielet, metriikka-arvot,...**
- ◇ **Vaatimukset: tehtävätyyppi, laajuus, kompleksisuus,...**
- ◇ **Organisatorinen ympäristö: johdon päätökset, henkilövalinnat, koulutus**
- ◇ **Päätökset liittyen ohjelmakoodiin: työvälineet, standardit, uudelleenkäyttö, ...**
- ◇ **Päätökset liittyen vaatimukseen: käytettävät prosessimallit, protoilu, inkrementaalinen kehitys, julkistukset, ...**
- ◇ **Päätökset liittyen em. pääkomponenttien vuorovaikutukseen: muutosoikeudet, muutosvastuut, muutosten koordinaatio, testauskäytänteet, ...**

YLLÄPITOON VAIKUTTAVAT TEKIJÄT (1):

- ◇ **Kyseessä on perinteisesti melko niukasti tutkittu (ICSM 1980-, JSM 1989-, IWPC 1992-, WCRE 1993-) systematisoitu ja opetettu aihealue**
- ◇ **Asenteet: arvostuksen puute, varautumisen puute, ...**
- ◇ **Yleinen laadunvarmistus kehityksen aikana: katselmukset, tarkastukset**
- ◇ **Systemaattinen palautteen kerääminen käyttäjiltä**
- ◇ **Systemaattinen regressiotestaus, testiaineistojen tallentaminen**
- ◇ **Standardien noudattaminen (ohjelmointikielet, käyttöliittymät, tietokannat, ...)**
- ◇ **Konfiguraationhallinta, muodolliset ja systemaattiset muutosprosessit, julkistusperusteinen muutosten toteuttaminen**
- ◇ **Ohjelmistojen laajuus ja kompleksisuus**
- ◇ **Redundanssi: monenkertainen koodi, uudelleenkäytön ohittaminen**

YLLÄPITOON VAIKUTTAVAT TEKIJÄT (2):

- ◇ **Pitkä käyttöikä: ohjelmiston “rapautuminen”**
- ◇ **Se, että vanhasta ohjelmistosta ei voida luopua (*legacy systems*): investoinnin säilyttäminen**
- ◇ **Dokumentaation laatu: kattavuus, oikeellisuus, ajantasaisuus, tiiviys, selkeys, yhtenäisyys, kehityshistoria, virhehistoria, ...**
- ◇ **Kommentoinnin laatu: funktiot, moduulit, liittymät, tietojoukot, määrittelyt, kompleksiset rakenteet**
- ◇ **Käytettävät ihmisresurssit: alkuperäiset kehittäjät ylläpitovastuussa/käytettävissä?**
- ◇ **Suunnittelun loogisuus ja suunnitteluperusteiden jäljitettävyys**
- ◇ **Suunnitteluperiaatteet: rakenteisuus, modulaarisuus, oliokeskeisyys, spekulatiivinen suunnittelu, laajennettavuuden varhainen huomionti**
- ◇ **Ohjelmointityylin vakiintuneisuus ja yhtenäisyys**
- ◇ **Kehitysvälineiden abstraktiotaso: Assembler, ... , sovelluskehittimet**

YLLÄPITOON VAIKUTTAVAT TEKIJÄT (3):

- ◇ **“Vanhat” ohjelmointikielet: Fortran, COBOL, C, ...**
- ◇ **Useiden ohjelmointikielien käyttö**
- ◇ **Ohjelmiston tyyppi: kaupallishallinnolliset-, sulautetut-, reaaliaikajärjestelmät, ...**
- ◇ **Sovellusalueen vakiintuneisuus, vaatimusten muutostiheys**
- ◇ **Sovellusalueen riippuvuus muuttuvasta ulkoisesta ympäristöstä**
- ◇ **Ylläpidettävyyden ohittaminen toteutustasolla: laajennettu prototyyppi, optimoinnit, viritelmät, ...**
- ◇ **Ohjelmiston yleisyyteen panostaminen: toimivuus eri laitteistoilla, käyttöjärjestelmissä, syöte/tulos tietoformaateilla, toimivuus erilaisilla tietorakenteilla/algoritmeilla**
- ◇ **Ohjelmiston joustavuuteen panostaminen: moduulien ja luokkien liittymien suunnittelu, ...**
- ◇ **Versioiden ja varianttien lukumäärä**

YLLÄPITOON VAIKUTTAVAT TEKIJÄT (4):

- ◇ **Toteutustason ylläpitoa tukevat ratkaisut:
jäljitysmekanismit, ehdollinen käänös, proseduurien
alku- ja loppuehdot, defensiivinen ohjelmointi, ...**
- ◇ **Apuvälineiden (CASE) käytettävyys (käytettävälle
laitealustalle, käyttöjärjestelmälle,
ohjelmointikielelle, ...)**

LEHMANIN LAIT (A):

◇ Empiirisesti todennettuja lainalaisuuksia

- 1) Jatkuvan muutoksen laki: “Ohjelman, jota käytetään jossakin tosimaailman ympäristössä täytyy muuttua, tai se tulee jatkuvasti vähemmän käyttökelpoiseksi siinä ympäristössä.”**
- 2) Lisääntyvän mutkikkuuden laki: “Kun ohjelma kehittyy ja muuttuu, sen rakenne pyrkii tulemaan monimutkaisemmaksi. Rakenteen säilyttämiseen ja yksinkertaistamiseen tarvitaan ylimääräistä työtä.”**
- 3) Suurten ohjelmien kehityksen laki: “Ohjelman kehittyminen (evoluutio) on itseänsäatelevä prosessi. Sellaiset ominaisuudet kuin: koko, julkistusten välinen aika ja havaittujen virheiden määrä pysyvät suunnilleen vakioisina järjestelmän jokaisessa julkistuksessa.”**

LEHMANIN LAIT (B):

- 4) Organisatorisen stabiiliuden laki: “Ohjelman kehitysnopeus on sen elinaikana suunnilleen vakio ja riippumaton järjestelmäkehitykseen varatuista resursseista.”**
- 5) Tuttuuden säilymisen laki: “Järjestelmän elinikänä inkrementaalinen muutos kussakin julkistuksessa on suunnilleen vakio.”**

KIIREISEN MUUTOSTYÖN PROSESSI (SOMMERVILLE):

- 1) Muutospyynnön saapuminen**
- 2) Vaikutusanalyysi**
- 3) Uuden version suunnittelu**
- 4) Muutosten toteutus ja validointi**
- 5) Uuden version julkaisu**

**SYSTEMAATTINEN MUUTOSTYÖ (SOMMERVILLE,
MARTIN & MCCLURE):**

- 1) Vaatimusmäärittelyn muutos: vaatimusten muodostus, täsmentäminen, validointi ja dokumentointi**
- 2) Järjestelmän komponenttien uudelleenmuokkauksen suunnittelu: oleellisten komponenttien ja tietorakenteiden tunnistaminen**
- 3) Oleellisten komponenttien yksityiskohtainen tulkinta ja ymmärtäminen**
- 4) Toteutus: ohjelmamuutokset**
- 5) Vaikutusanalyysi ja testaus: alkuperäiset testit, mahdollisesti uusia testejä, ...**
- 6) Muutosten dokumentointi, ...**

YLLÄPIDON “GENEERISET” OSATEHTÄVÄT:

- ◇ **Pieniä aktiviteetteja, joista laajemmat ylläpitoprosessit tyypillisesti koostuvat**
- ◇ **Tunnistamistehtävät: häiriöt ohjelman toiminnassa, vaatimukset ja dokumentit, jotka liittyvät tiettyyn komponenttiin, ...**
- ◇ **Lukeminen ja tulkinta: ohjelmat, kommentit, dokumentaatio, testidata, ...**
- ◇ **Suunnittelu: tarvittavat ohjelmamuutokset**
- ◇ **Käsitteellinen osittaminen: kohteena laajat ohjelmat**
- ◇ **Etsimistehtävät: muutettavat ohjelmaosat, virheen aiheuttavat ohjelmaosat, muutokseen vaikuttavat ohjelmaosat, ...**
- ◇ **Varsinaiset muutostehtävät: koodi, dokumentaatio, ...**

OHJELMAMUUTOSTEN TEON YLEISPERIAATTEITA

(MARTIN & MCCLURE):

- ◇ **Konservatiivisuus: pienimpien mahdollisten muutosten teko**
- ◇ **Muutos tulee toteuttaa oikein**
- ◇ **Ohjelmiston laatu ei saa heikentyä**
- ◇ **Ohjelmointityyli ja kokonaisuuden toiminnallisuus tulee säilyä**
- ◇ **Tulevat ohjelmamuutokset tulee voida tehdä helposti**
- ◇ **Ohjelman käyttäjiin ei tule kohdistua negatiivisia seuraamuksia**

**OHJELMAMUUTOSTEN TEON YKSITYISKOHTAISIA
OHJEITA (YOURDON, GLASS & NOISEAUX, ...):**

- ◇ **Kontrollirakenteeseen tutustuminen jo ennen
 pieniäkin muutoksia**
- ◇ **Koodin rakenteistaminen**
- ◇ **Uusien paikallisten muuttujien käyttöönotto
 (vanhojen käytön sijasta)**
- ◇ **Muutoskirjanpito**
- ◇ **Defensiivinen ohjelmointi**
- ◇ **Vanhan version säilytys**

VAIKUTUSANALYYSI:

- ◇ **Negatiivisten sivuvaikutusten tunnistamista**
- ◇ **Koodiperusteiset-: käytössä olevien elementtien poistot, muistitarpeen muutos, nopeuden muutos, ajoitusmuutos, ...**
- ◇ **Tietoperusteiset-: lippumuuttujan muutos, tietueen koon muutos, ...**
- ◇ **Dokumentointiin liittyvät-: oletustoiminnallisuuden muutos, lisävaatimukset käyttäjälle, ...**

TIETOTARPEET:

- ◇ **Jostakin tietolähteestä saatavia tietoelementtejä, joita ylläpitotehtävän menestyksellinen suorittaminen edellyttää**

TIETOLÄHTEET:

- ◇ **Ohjelmakoodin staattisen analyysin tulokset**
- ◇ **Ohjelmakoodin dynaamisen analyysin tulokset**
- ◇ **Dokumentaation (ml. muutoshistoriat) analyysin tulokset**
- ◇ **Oppikirjat yms. yleinen materiaali**
- ◇ **Tieto ylläpitäjän omasta dynaamisesta toiminnasta (toimintahistoriat)**
- ◇ **Työtoverit**

TIETOTARVETUTKIMUKSET (VON MAYRHAUSER YM.):

- ◇ **Havainnoivia empiirisiä kenttätutkimuksia**
- ◇ **Edustavat tarkinta nykyistä tutkimustietoa C-kielisten ohjelmien ylläpitäjien tyypillisesti tarvitsemasta tiedosta**
- ◇ **Kohdentuvat edustavaan ylläpidon osaan**
- ◇ **Tehtävät: todellisia työtehtäviä kattaen korjaavan-, mukauttavan- ja viimeistelevän ylläpidon**
- ◇ **Ohjelmat: tuotantokäytössä olevia vähintään 40 KLOC:n C-kielisiä ohjelmia**
- ◇ **Ylläpitäjät: pieni joukko ohjelmistotyön ammattilaisia**

EHKÄISEVÄ YLLÄPITO:

- ◇ **Uudelleenkonstruointi (*re-engineering*): järjestelmän tutkimista ja muuttamista tavoitteena muuttaa se (usein myös toiminnallisesti) uuteen muotoon**
- ◇ **Uudelleenrakenteistaminen (*restructuring*): järjestelmän sisäisen esitysmuodon muuttamista vakioabstraktiotasolla (järjestelmän ulkoinen käyttäytyminen ei myöskään muutu)**
- ◇ **Uudelleendokumentointi (*redocumentation*): semanttisesti alkuperäistä dokumenttia vastaavan havainnollisemman kuvauksen luontia**

KORJAAVAAN YLLÄPITOON LIITTYVIÄ KYSYMYKSIÄ:

- ◇ **Miksi tietyn tyyppiset virheet ovat vaikeita korjata, mitkä ovat taustalla olevat yleiset syyt?**
- ◇ **Mitkä ovat virheiden tavalliset, yksityiskohtaiset, syyt?**
- ◇ **Miten virheet tavallisesti löydetään tai ongelmat ratkaistaan?**
- ◇ **Mitkä ovat eniten virheitä aiheuttavia ohjelmarakenteita?**

VUOSI 2000 -ONGELMA (Y2K):

- ◇ **Erinomainen esimerkki korjaavan ylläpidon tehtävästä**
- ◇ **Syy: 'ppkkvv' -muoto**
- ◇ **Ilmeneminen: vuosien 1999 ja 2000 vaihteessa**
- ◇ **Merkitys: kaikkien aikojen kallein yksittäinen tekninen ongelma**
- ◇ **Virheellinen toiminta: esim. vuosi 1900 oletus**
- ◇ **Mahdolliset seuraukset: ohjelmistojen toiminta ennalta-arvaamattomalla tavalla**
- ◇ **Suunnitteluvirheet: päivämäärän käsittelyä ei oltu koteloitu selkeän rajapinnan kautta käsiteltäväksi**
- ◇ **Ongelma oli hyvin tiedostettu ennen vuotta 2000**
- ◇ **Ongelman suuruusluokan arviointi oli etukäteen vaikeaa (arviot välittömistä ja välillisistä kustannuksista vaihtelivat välillä 300-1600 mrd US\$)**
- ◇ **Ongelma ratkaistiin hyvin onnistuneesti panostamalla virheen korjaukseen**
- ◇ **Käytännössä, esim. USA:ssa liittovaltio käytti viiden vuoden aikana 8.38 mrd US\$**

VUOSI 2000, TOTEUTUNEITA HÄIRIÖITÄ:

- ◇ **Japanissa, joissakin ydinvoimaloissa häiriöitä ei-kriittisissä osissa**
- ◇ **Ruotsissa, joissakin Uppsalan ja Lundin sairaaloissa sydänvalvontalaitteissa virheellistä tietoa**
- ◇ **USA:ssa, joissakin vakoilusatelliittien tuottamaa informaatiota käsittelevissä tietokoneissa ongelmia**
- ◇ **Maailmanlaajuisesti n. 900:ssa kirjastossa käytettävä VTLS-järjestelmä poisti vanhentuneiksi tulkitsemiaan kirjavarauksia**

TUTKIMUS SUOMEN VUOSI 2000 -TILANTEESTA:

- ◇ **Kohde: 33 Suomen suurinta yritystä + 5 suurta julkista organisaatiota**
- ◇ **Tulos: 68 %:lla ei ollut vuodenvaihteessa ongelmia, 32 %:lla oli pieniä häiriöitä**
- ◇ **Panostukset: keskimäärin n. 8 miljoonaa euroa/yritys**
- ◇ **Nokia: n. 75 miljoonaa euroa**

KARKAUSPÄIVÄ -ONGELMA:

- ◇ **Karkauspäivä (ylimääräinen helmikuun viimeinen päivä): lisätään joka 4. vuosi, poikkeus: ei sadalla jaollisina vuosina, poikkeuksen poikkeus: lisätään 400:lla jaollisiin (vuosi 2000 oli tällainen poikkeus)**
- ◇ **Jotkin ohjelmistot ovat joskus aiemmin tulkinneet karkauspäivän maaliskuun 1:ksi tai helmikuun 30:ksi**
- ◇ **Vuonna 2000: ei juuri ongelmia, koska karkauspäivän käsittely oli jo testattu vuosi 2000 -ongelmaan liittyen**

OHJELMIEN TULKINTA JA YMMÄRTÄMINEN:

- ◇ **Tärkeää erityisesti ylläpitovaiheessa**
- ◇ **Ymmärryksen aikaansaanti liittyy kiinteästi ylläpitoprosessiin**
- ◇ **Ylläpitäjän tavoitteena on usein ohjelman alkuperäisten tekijöiden ajatusmallien uudelleenluonti mielessään**
- ◇ **Riittävä ymmärrys on välttämätön edellytys hallituille ohjelmamuutoksille**
- ◇ **Keskeistä: ohjelman tarkastelun abstraktiotason nosto ja tilannekohtaisesti oleellisten tekijöiden (komponentit, tietorakenteet jne.) tunnistaminen**

OHJELMIEN YMMÄRTÄMISEN TEORIAT:

- ◇ **On esitetty monia teorioita (Brooks, Letovsky & Soloway, Shneiderman, Pennington, von Mayhauser & Vans, ...)**
- ◇ **Eri teorit painottavat ymmärtämisprosessin eri aspekteja**
- ◇ **Teorioita ei ole pystytty validoimaan kovinkaan hyvin**

OHJELMIEN YMMÄRTÄMISEEN KÄSITTEET:

- ◇ **Mentaalinen malli:** ylläpitäjän mielessä oleva kuvaus tehtävään liittyvistä oleellisista tekijöistä
- ◇ **Suunnitteluperuste:** toteutuksen taustalla oleva syy tehtyihin valintoihin
- ◇ **Ohjelmasuunnitelma:** vakiotyyppinen toimintajono, jolla on tapana saada aikaan tietty tarkoite
- ◇ **Ei-paikallinen ohjelmasuunnitelma:** hajautunut ohjelmasuunnitelma
- ◇ **Lohko:** ymmärtämisen kannalta merkityksellinen ohjelmaosa
- ◇ **Suunnannäyttäjä:** helposti tunnistettava ohjelmaosa, joka voi toimia lähtökohtana prosessin etenemiselle
- ◇ **Kohdealueen sääntö:** yleisesti noudatettava periaate, jota voidaan käyttää perustana prosessille
- ◇ **Ymmärtämisstrategia:** ohjelman selaustapa, jolla tarvittava ymmärrys tehokkaasti saavutetaan
- ◇ **Systemaattinen strategia vs *ad hoc* strategia**
- ◇ **Osittava vs kokoava strategia**
- ◇ **Opportunistinen strategia**

INTEGROITU YMMÄRTÄMISMALLI (VON MAYRHAUSER & VANS):

- ◇ **Integroitu malli, joka kuvaa monissa muissa malleissa esitettyjä asioita**
- ◇ **Neljä pääosaa/tulosta: osittava malli, ohjelmamalli, tilannekohtainen malli ja tietämuskanta**

OSITTAVA MALLI (*TOP-DOWN MODEL*):

- ◇ **Mallin yläosa**
- ◇ **Järjestelmä jaetaan käsitteellisesti yleisesti ymmärrettäviin osiin**
- ◇ **Mallin muodostuksessa käytetään apuna sovellusalueen tietämystä**
- ◇ **Sovelletaan erityisesti, kun ohjelma on suhteellisen tuttu**
- ◇ **Ymmärtämisstrategia: päämääräsuuntautunut, opportunistinen, hypoteesivetoinen**

OHJELMAMALLI (*PROGRAM MODEL*):

- ◇ **Mallin vasen puoli**
- ◇ **Kuvaa ohjelman kontrollivuota**
- ◇ **Sovelletaan erityisesti, kun ohjelma on täysin tuntematon**
- ◇ **Ymmärtämisstrategia: tyypillisesti kokoava, systemaattinen, seurataan ohjelman kontrolli- ja tietovirtoja**

TILANNEKOHTAINEN MALLI (*SITUATION MODEL*):

- ◇ **Mallin oikea puoli**
- ◇ **Samankaltainen, kuin ohjelmamalli**
- ◇ **Ymmärtämisstrategia: systemaattinen tai opportunistinen**

TIETÄMYSKANTA (*KNOWLEDGE BASE*):

- ◇ **Mallin keskiosa**
- ◇ **Ylläpitäjän pitkäkestoisen muistin relevantti sisältö**
- ◇ **Tietämys liittyy aiemmin mainittuihin käsitteisiin (esim. ohjelmasuunnitelmat, kohdealueen säännöt ja suunnannäyttäjät)**

TIETÄMYSTYYPIT:

- ◇ **Sovellusalue-tietämys**
- ◇ **Toteutustekniikkatietämys**
- ◇ **Strategiset suunnitelmat: määrittävät ohjelman yleisen toimintatavan ja kieliriippumattomia asioita**
- ◇ **Taktiset suunnitelmat: määrittävät paikallisia, kieliriippuvia ongelmanratkaisuun liittyviä asioita**
- ◇ **Toteutussuunnitelmat: kieliriippuvia suunnitelmia, jotka toteuttavat taktisen tason suunnitelmat**

YLLÄPIDETTÄVYYDEN METRIIKAT (SOMMERVILLE, BOEGH):

- ◇ **Perusoletus: kompleksisuus heikentää ylläpidettävyyttä**
- ◇ **Ohjelmarivien määrä**
- ◇ **Moduulien lukumäärä**
- ◇ **Keskimääräinen moduulien koko**
- ◇ **Kontrollikompleksisuus: ohjelman ehdollisten lauseiden lukumäärä**
- ◇ **Tietokompleksisuus: tietorakenteiden ja komponenttien välisten liittymien kompleksisuus**
- ◇ **Integraatiotestauksen aikana korjattujen virheiden seurauksena muuttuneiden moduulien lukumäärä**
- ◇ **Kytkeäaste: moduulien väliset riippuvuudet**
- ◇ **Dokumentation suhteellinen määrä**
- ◇ **Kommenttien suhteellinen määrä**
- ◇ **Käyttäjien vuorovaikutuksen määrä**
- ◇ **Aika- ja tilavaatimusten tiukkuus**
- ◇ **McCabe:n, Halstead'in, Kafura:n & Reddy:n esittämät kompleksisuusmitat**

SUUNNITTELUVAIHEEN METRIIKOITA OLIO-OHJELMIEN YLLÄPIDETTÄVYYDELLE (LI YM.):

- ◇ **Luokan sijainti periytymishierarkiassa suhteessa juuriluokkaan (DIT)**
- ◇ **Paikallisten metodien määrä + ei-paikalliset metodit, joita paikalliset kutsuvat (RFC)**
- ◇ **Paikallisten metodien jakautumattomien joukkojen määrä (LCOM)**
- ◇ **Toisen luokan instanssien esittelyjen lukumäärä (DAC)**
- ◇ **Luokan metodien lukumäärä (NOM)**

YLLÄPIDETTÄVYYDEN PROSESSIMETRIIKAT (SOMMERVILLE):

- ◇ **Korjaavaan ylläpitoon liittyvien korjauspyyntöjen lukumäärä**
- ◇ **Vaikutusanalyysiin keskimäärin tarvittava aika**
- ◇ **Muutostarpeen toteuttamiseen keskimäärin tarvittava aika**
- ◇ **Huomattavien muutostarpeiden lukumäärä**

YLLÄPIDON KUSTANNUSTEN ARVIOINTI:

- ◇ **Koska kustannusvaikutus on yleensä suuri, asia on tärkeä**
- ◇ **Tarkka arviointi on kuitenkin yleensä vaikeaa**
- ◇ **Systemaattisia lähestymistapoja on kehitetty perustuen tietoon aikaisemmista samankaltaisista projekteista (sama sovellusalue, samantyyppinen järjestelmä, samankaltainen kehitysprosessi)**

COCOMO:N SOVELTAMINEN YLLÄPITOKUSTANNUSTEN ARVIOINTIIN (BOEHM):

- ◇ **ACT (*Annual Change Traffic*): ohjelmiston lähdekoodiosuus, johon kohdistuu lisäyksiä tai muutoksia**
- ◇ **SDT (*Software Development Time*): ohjelmiston kehittämisen resurssitarve (htkk.)**
- ◇ **AME (*Annual Maintenance Effort*): vuosittainen ylläpitotyön tarve = $ACT * SDT$**
- ◇ **Antaa karkean arvion kustannuksille, oletuksena lineaarinen riippuvuus**
- ◇ **Malli täytyy kalibroida ottaen huomioon kehittäjäorganisaation erityispiirteet**
- ◇ **Myös COCOMO:n välitason mallin esittämiä lisätekijöitä voidaan pyrkiä ottamaan huomioon**
- ◇ **Lisätekijöiden soveltuvuus ylläpidon arviointiin on rajallinen**
- ◇ **Ylläpitoon vaikuttaa myös muita em. tekijöitä, joiden tarkka arviointi on vaikeaa**

YLLÄPIDON TUEN TEKNIIKAT

KÄÄNTEISTEKNIikka (*REVERSE ENGINEERING*) (CROSS YM.)

- ◇ **Järjestelmän sisältämien komponenttien ja suhteiden tunnistaminen**
- ◇ **Niiden esittäminen uudessa, tyypillisesti abstrahoidussa muodossa**

KOKOONPANON PALAUTUS (*DESIGN RECOVERY*):

- ◇ **Apuna käytetään ulkopuolista dataa ja sumeaa päättelyä**
- ◇ **Tuloksena mahdollisesti suunnitteludokumentin runko, ER-kaavioita jne.**
- ◇ **Historiallinen tausta: kilpailijoiden valmistamien laitteistojen rakenteen selvittäminen**

KÄÄNTEISTEKNIKKATYÖKALUN OSAT:

- ◇ **Taustaosa: jäsentäjät, analysaattorit, tietovarasto: jäsenpuu, symbolitaulu, ohjelmariippuvuusverkot tms.**
- ◇ **Edustaosa: vuorovaikutusmekanismit, kyselykieli tms., tiedon esittäminen käyttäjälle, visualisointimekanismit**

TIETOKYSELYT:

- ◇ **Lähtökohta: tietotarpeet**
- ◇ **Spesifiointitavat: esimääritellyt valinnat, yksinkertainen QBE-tyyppinen liittymä, täydellinen kyselykieli, esim. SCA**

SCA: SOURCE CODE ALGEBRA (PAUL & PRAKASH)

- ◇ **Relaatioalgebra + transitiivinen sulkeuma + jono-operaattorit**
- ◇ **Hyvät puolet: ilmaisuvoima, suhtellisen kompaktit kyselyt**
- ◇ **Huonot puolet: kyselyjen muodostus vie aikaa**

OHJELMIEN VISUALISOINTI:

- ◇ **Koodin rakenne: muotoilijat, karsinta, laajennus, zoomaus, värikoodaus, kalansilmänäkymät, synkronoitu vieritys**
- ◇ **Ohjelmiston rakenne: erilliset näkymät, suunnatut verkot**
- ◇ **Ohjelmiston staattiset ominaisuudet: metriikka-arvojen värikoodaus**
- ◇ **Ohjelmiston dynaamiset ominaisuudet: suoritusprofiili, algoritmien animointi, tietorakenteiden sisällön animointi**

OHJELMISTON STAATTISET JA DYNAAMISET

OMINAISUUDET (BALL & EICK):

- ◇ **Kukin koodin osa kuvataan pikselillä**
- ◇ **Pikselin väri kuvaa ominaisuuden voimakkuutta**
- ◇ **Mahdollisia ominaisuuksia: koodin ikä, muutosaste, korjausaste, virheaste, käännöksen ehdollisuustyyppi, kompleksisuus, sisäkkäisyysaste, käyttöaste, ...**
- ◇ **Esim. koodin ikä: uusimmat (punainen), vanhimmat (sininen), ... (52 tiedostoa, 15 KLOC)**
- ◇ **Dynaamiset ominaisuudet: aktiiviset ohjelmarivit (punainen, keltainen)**
- ◇ **Ohjelmaviipaleet (keltainen)**

OHJELMISTON RAKENNE (IMAGIX 4D):

- ◇ **3-ulotteinen kutsukaavio: pyöritys, zoomaus, karkeustason valinta**
- ◇ **2-ulotteinen vuokaavio: testatut vaihtoehtoiset kontrollivuon reitit**

OHJELMIEN VIIPALOINTI:

- ◇ **Mark Weiser (1982)**
- ◇ **Viipale: ohjelman käskyjen osajoukko**
- ◇ **Viipalointi: prosessi, jolla poimitaan ohjelmasta mielenkiintoiset käskyt**
- ◇ **Lähtökohta: viipalointikriteeri (tavallisesti muuttajan esiintymä)**
- ◇ **Taaksepäin viipalointi vs eteenpäin viipalointi**
- ◇ **Staattinen vs dynaaminen viipalointi**
- ◇ **Intraproseduraalinen vs interproseduraalinen (täydellinen) viipalointi**

TAAKSEPÄIN VIIPALOINTI:

- ◇ **Perinteinen muoto**
- ◇ **Sovelluskohde: virheenjäljitys**
- ◇ **Viipalointi etenee takaperin kontrollivirtaa pitkin**

ETEENPÄIN VIIPALOINTI:

- ◇ **Sovelluskohde: vaikutusanalyysi**
- ◇ **Viipalointi etenee eteenpäin kontrollivirtaa pitkin**

STAATTINEN VIIPALOINTI:

- ◇ **Kohde: ohjelman yleisen käyttäytymisen tarkastelu**
- ◇ **Konservatiivisuus: viipaleeseen sisällytetään kaikki mahdollisesti oleellinen**
- ◇ **Viipaleet ovat suhteellisen laajoja**

DYNAAMINEN VIIPALOINTI:

- ◇ **Kohde: tietyn ajokerran tarkastelu**
- ◇ **Muuttujien todelliset arvot otetaan huomioon**

STAATTINEN, ITERATIIVINEN TIETOVIRTAYHTÄLÖIDEN

RATKAISEMINEN:

- ◇ **Manuaalinen vs automaattinen**
- ◇ **Aputietorakenteet: jäsenyspuu, symbolitaulu**
- ◇ **Pidetään kirjaa oleellisista muuttujista (RVS):
symbolitauluviittaukset**
- ◇ **Analysoitavan lauseen liittäminen viipaleeseen: jos
sen osana on muuttuja, joka kuuluu/vaikuttaa
RVS:ään**
- ◇ **Oletus: C-kielen kaltaisen imperatiivisen kielen
analysointi**
- ◇ **Merkitykselliset lauseet: sijoitus-, ehto-, toisto-,
parametrinvälitys**
- ◇ **Toistolauseet: RVS:n vakiintuminen**

INTERPROSEDURAALINEN VIIPALOINTI:

- ◇ **Viipalointikriteeri => lähtökohtafunktio**
- ◇ **Alasviipalointi: lähtökohtafunktion kautta kutsuttujen funktioiden viipalointi**
- ◇ **Ylösviipalointi: niiden funktioiden viipalointi, joiden kautta voidaan päätyä lähtökohtafunktioon**
- ◇ **Staattisen viipaloinnin käytettävyyden rajoite: kutsukontekstiongelman**
- ◇ **Kutsutasojen määrän kasvaessa yleensä vaihtoehtoisten kutsupolkujen määrä kasvaa nopeasti => analyysi on hidasta**
- ◇ **Tehostamiskeinoja: esimääritellyt staattiset apurakenteet, ohjelmariippuvuusverkot, laajennetut viipalointikriteerit**

VIIPALEIDEN ESITTÄMINEN KÄYTTÄJILLE:

- ◇ **Erillinen näkymä (yhtenäinen kokonaisuus)**
- ◇ **Korostettuna ohjelmatekstin joukossa (konteksti näkyvillä)**
- ◇ **Välimuoto: korostettu linkitetty ohjelmateksti (HyperSoft)**
- ◇ **Ohjelmariippuvuuksien esittäminen**

SYSTEMAATTINEN KORJAAVA YLLÄPITO (JAMBOR-SADEGHI YM.):

- ◇ **Ongelmia: virheraporttien abstraktius ja monimerkityksisyys, tarvittavien tiedon esitysmuotojen heterogeenisuus, prosessin heuristisuus**
- ◇ **Korjaavan ylläpidon prosessimalli**
- ◇ **Tavoitteet: tietojen yhtenäinen esittäminen, abstraktien näkymien muodostaminen, assosiaatioiden eksplikointi**
- ◇ **Edellytykset: ohjelmiston staattinen rakenteellinen ositus, dynaaminen suorituspolkujen selvittäminen**
- ◇ **SAMS: puoliautomaattinen, kokeellinen apuvälineohjelma**
- ◇ **SAMS tuottaa oliomallit/tietokannat: rakenteellinen-, toiminnallinen-, suorituspolku-, virhe-, testitapaus-**
- ◇ **Malli kuvaa prosessin vaiheet ja niihin tyypillisesti liittyvät tietotarpeet (joita apuväline voi tukea)**
- ◇ **Muita käyttökohteita: vaikutusanalyysi, regressiotestaus (mukauttava ylläpito)**

HYPERSOFT (KOSKINEN):

- ◇ **Sovellettu tekniikka: automaattinen väliaikaisen hypertekstin muodostus**
- ◇ **Sovelluskohde: ohjelmien ylläpidon ja ohjelmien ymmärtämisen tuki**
- ◇ **Projekti: TEKES, Nokia, TietoEnator, Novo Group (1994-1997)**
- ◇ **Toteutus: kokeellinen ylläpidon tuen käännteistekniikkatyökalu (ANSI-C, ESQL-tuki)**
- ◇ **Tukimuodot: hypertekstinäkymät + graafiset näkymät**
- ◇ **Hyperteksti: mahdollistaa rakenteellisen (ohjelma)tekstin epälineaarisen selauksen**
- ◇ **Graafiset näkymät: mahdollistavat ohjelmien abstraktin kuvauksen ja nopean siirtymän vastaaviin kohtiin ohjelmatekstissä**
- ◇ **Tiedonhaun tuen perusta: hakurakenteet (THAS = *Transient Hypertextual Access Structure*)**
- ◇ **Toimivuuden validointi: empiirinen testaus**

TEOREETTINEN TAUSTA:

- ◇ **Hypertekstin muodostuksen kultaiset säännöt (laajuus, ositettavuus, riippuvuudet, käyttäjä tarvitsee kerralla vain pientä osaa)**
- ◇ **Hyperteksti koostuu solmuista ja linkeistä**
- ◇ **Solmut vastaavat ohjelmaosia (ohjelmointikielen syntaktiset tyypit)**
- ◇ **Linkit vastaavat ohjelmariippuvuuksia (esim. kutsu-, tietovirta-, kontrollivirta-, ...)**
- ◇ **Kielioppiperusteisuus: jäsennysooesitysmuoto**
- ◇ **Joustavuus: tilannekohtaisten tietotarpeiden suoraviivainen tyydyttäminen**
- ◇ **Hypertekstin väliaikaisuus: ohjelmamuutosseurausten, hypertekstin ylläpito-ongelmien välttäminen**

HYPERSOFT-MALLIN TASOT:

- ◇ **Lähdekooditaso: ohjelmakoodin lineaarinen esitys ohjelmatiedostoissa**
- ◇ **Syntaktinen taso: jäsennysooesitys ja -operaatiot**
- ◇ **Hakurakennetaso: hypertekstirakenteet ja -operaatiot**
- ◇ **Käyttöliittymätaso: ohjelmatekstin, hypertekstin, graafisten näkymien esittäminen käyttäjille, käyttäjän ja järjestelmän vuorovaikutus**

HYPERSOFT (v. 1.0, 1997):

- ◇ **Tuetut ohjelmointikielet: ANSI-C, ESQL**
- ◇ **Käyttöjärjestelmä: Microsoft Windows
3.01/95/NT/2000**
- ◇ **Perusosat: staattinen ohjelma-analysointilaite,
hypertekstirakenteiden tuottaja, käyttöliittymä,
integroitu tekstieditori (PFE)**
- ◇ **Tuetut hakurakennetyypit: esiintymälistat,
kutsukaaviot, ohjelmaviipaleet**
- ◇ **Tuetut näkymätyypit: ohjelmatekstin lineaarinen
esitys, sulautettu hypertekstinäkymä, rakenteinen
karttanäkymä, graafinen
funktio/moduuliriippuvuusnäkyminen**

HYPERSOFT:IN KÄYTTÖ (1):

- ◇ **Valmistelevat toimenpiteet: projektin valinta, polkumäärittelyt**
- ◇ **Projektin valinta: ‘*File/Project Manager*’**
- ◇ **Aktiivisen projektin sisältö: ‘*View/Project File*’**
- ◇ **Staattisen ohjelmatietokannan muodostus: ‘*File/Analyze Project*’**
- ◇ **Virhelokin tarkastelu, mahdollisten virheiden korjaus, ... (kuten kääntäjiä käytettäessä)**

HYPERSOFT:IN KÄYTTÖ (2):

- ◇ **Hakurakenteen muodostus: rakenteen alkukriteerin valinta tekstikohdistimella + ‘*Query/...*’**
- ◇ **Näkymien muodostus (jo muodostetulle hakurakenteelle): ‘*View/...*’**
- ◇ **Aiemmin muodostettujen hakurakenteiden aktivointi: ‘*View/Select access structure*’**
- ◇ **Hakurakenteen alku- eli kotisolmu: F5 (tai vastaava talo-ikoni)**
- ◇ **Navigointivalintojen peruutustoiminto: F6 (tai vastaava U-nuoli-ikoni)**
- ◇ **Historialista (käyttäjän valitsemat polut): ‘*Navigation/History list*’**
- ◇ **Linkkien graafinen esitys ohjelmatekstin päällä: ‘*Navigation/Show all links*’**
- ◇ **Integroidun ohjelmaeditorin (PFE) käynnistys: ‘*View/Structured Map for Editor*’**
- ◇ **Ohjeet, toiminnallisuuksien ja rajoitteiden kuvaus: ‘*c:\hsoft\help\hsoft.txt*’**