


Ohjelmistotuotanto

Ohjelmistosuunnittelu 1

1




Suunnittelu

- Suunnittelu:
 - asiakaslähtöisten vaatimusten muuntaminen teknisiä mahdollisuuksia tehokkaasti hyödyntäväksi ratkaisuksi
 - luova, asiantuntemusta vaativa prosessi
- Pääkohteet:
 - ohjelmiston yleisrakenne ja toimintaperiaatteet
 - tietorakenteet (ulkoiset, sisäiset)
 - algoritmit, toiminnot
 - käyttöliittymä

© Harri Laine, Jukka Paakki

2




Suunnittelu

- Suunnittelun tuloksena on tekninen malli järjestelmästä, joka aiotaan tehdä
- Kaksi hallinnollista kokonaisuutta: yleissuunnittelu ja yksityiskohtainen suunnittelu
- yleissuunnittelu
(*product design / system design / preliminary design*)
 - järjestelmän kokonaisrakenne
 - yhteiset tietorakenteet
 - jako pääkomponentteihin (alijärjestelmät, moduulit)
 - pääkomponenttien välisen kommunikoinnin periaatteet
 - ohjelmistoarkkitehtuuri

© Harri Laine, Jukka Paakki

3




Suunnittelu

- yksityiskohtainen suunnittelu
(*detailed design / program design / module design*)
 - kunkin komponentin sisäinen rakenne
 - algoritmit
 - rakenneosien yhteistyö
- Suunnitteludokumentti syntyy usein vastaavasti kahdessa vaiheessa:
 - yleissuunnitelma (arkkitehtuuri)
 - komponenttisuunnitelmat

© Harri Laine, Jukka Paakki

4




Suunnittelu

- *Tietosuunnittelu* (data design) muuntaa tietosisältömallin tietorakenteiksi
 - ulkoiset tietorakenteet (tietokantasuunnittelu)
 - tietokantakaaviot
 - sisäiset tietorakenteet
 - luokkakaaviot
- *Arkkitehtuurisuunnittelu* (architecture design) määrittelee ohjelmiston korkean tason komponentit ja niiden väliset suhteet
 - luokkakaaviot, olioiden yhteistyökaaviot, arkkitehtuurimallit, kehykset

© Harri Laine, Jukka Paakki

5



Suunnittelu

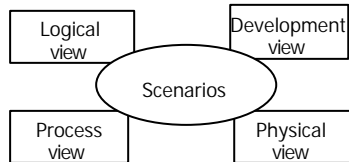
- *Liittymäsuunnittelu* (interface design) määrittelee, miten ohjelmiston ulkoiset liittymät toimivat
 - käyttöliittymä
 - laitteistoliittymät
 - tietoliikenne
- *Toimintasuunnittelu* (procedural design) kuvaa yksityiskohtaisesti toiminnallisuuden liittäminen arkkitehtuurisuunnitelmassa tunnistettuihin komponentteihin

© Harri Laine, Jukka Paakki

6

Arkkitehtuuri - ohjelmiston yleisrakenne

- P. Kruchten: The 4+1 View Model of Architecture, IEEE Software, Nov 1995, pp 42-50.
- Arkkitehtuuria on on syytä tarkastella eri näkökulmista



© Harri Laine, Jukka Paakki

7

Arkkitehtuuri - ohjelmiston yleisrakenne

- Logical view (looginen rakenne, sisältömalli)
 - esittää järjestelmän loogiset pääosat
 - alijärjestelmät, tärkeimmät luokat ja niiden väliset yhteydet
 - staattiset rakenteet, liittymät
 - yhteistoiminta osien välillä
 - tavoitteena on kuvan välittäminen sisällöstä kokonaisuutena
 - kuvaustekniikat: luokkakaavio, oliokaavio, yhteistyökaavio, tietovirtakaavio

© Harri Laine, Jukka Paakki

8

Arkkitehtuuri - ohjelmiston yleisrakenne

- Process view (toiminnan organisointi)
 - esittää kontrollin kulun järjestelmässä,
 - kuvaa prosessit, säikeet, rinnakkaisuuden
 - voidaan suhteuttaa sisältömalliin esimerkiksi määrittelemällä osajärjestelmänä
 - samasta osajärjestelmästä (prosessirakenteesta) voi olla useita instancesja (prosesseja)
 - osajärjestelmien kytkennät ja yhteistyö voidaan kuvata esim. (luokkakaavioilla, oliokaavioilla,) yhteistyö- ja sekvenssikaavioilla, tietovirtakaavioilla

© Harri Laine, Jukka Paakki

9

Arkkitehtuuri - ohjelmiston yleisrakenne

- Development view (ohjelmarakenne, toteutusnäkökulma)
 - ohjelmakomponenttien paketointi ja jako tiedostoihin
 - ohjelmistotekninen rakenne
 - minkälaisiksi kokonaisuuksiksi ohjelmisto pilkotaan?
 - kirjastot
 - komponentit
 - kehykset
 - millä perusteilla kootaan (kerrokset,...)
 - osien yhteenliittyminen voidaan kuvata luokka-, olio- ja komponenttikaavioilla tai moduulirakenteena
 - UML:ssä myös pakkauskaavio (package diagram)

© Harri Laine, Jukka Paakki

10

Arkkitehtuuri - ohjelmiston yleisrakenne

- Physical view (laitteistosijoittelu, hajautus)
 - ohjelmiston sijoittelu verkkoon ja laitteistoon
 - ohjelmiston ja prosessien sijoittelu
 - UML:ssä tarjolla erillinen kaavio (deployment diagram)
 - Sijoitettavia:
 - prosessit - prosessisijoittelu
 - ohjelmat - ohjelmasijoittelu
 - esim. WWW-sovelluksissa komponentti (esim. Java-sovelma) voi olla sijoitettu palvelimelle, mutta sitä suoritetaan työasemassa

© Harri Laine, Jukka Paakki

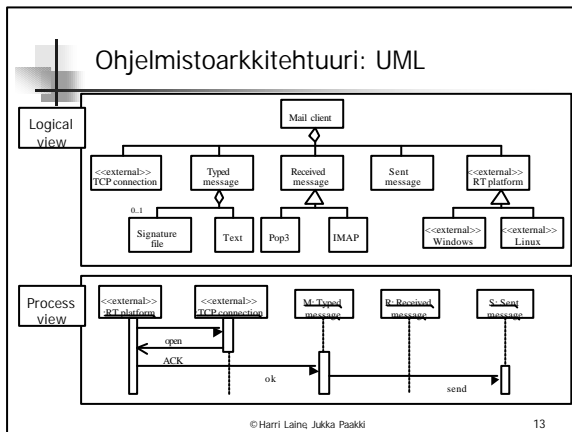
11

Arkkitehtuuri - ohjelmiston yleisrakenne

- Scenarios (skenaariot, käyttönäkökulma)
 - tyypillisen käytön esimerkitapauksia
 - kytkennät erityisesti sisältömalliin ja prosessimalliin
 - yhteistyö- ja sekvenssikaavioit keskeisiä (UML: collaboration diagram, sequence diagram)

© Harri Laine, Jukka Paakki

12

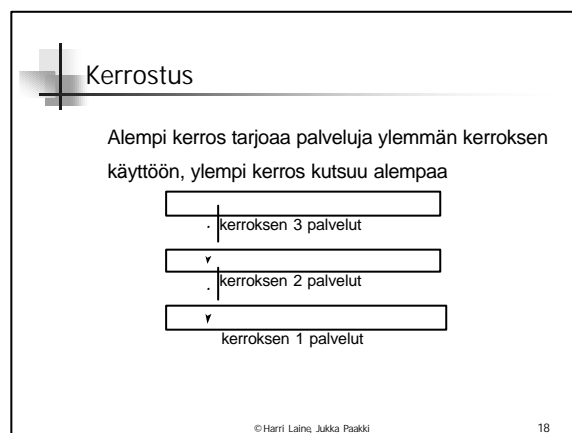


- ### Suunnittelu: osiin jakamista ja osien kytkemistä
- Suunnittelussa kokonaisuus jaetaan osiin
 - Jakamisella pyritään saavuttamaan:
 - toiminnallinen itsenäisyys
 - helpompi tehdä, testata ja ylläpitää
 - työnjako
 - ohjelmistoprojektissa
 - järjestelmän toiminnassa
 - käsiteltävyys
 - sopiva tarkkuustaso
 - sopiva laajuus
- © Harri Laine, Jukka Paakki 14

- ### Suunnittelu: osiin jakamista ja osien kytkemistä
- Jakamisessa ei ole yksikäsitteistä "parasta" ratkaisua
 - jako perustuu johonkin näkökulmaan, jonka mukaan tietyt palvelut ja tiedot kuuluvat samaan kokonaisuuteen
 - jakoperusteena voi olla:
 - käyttäjä, toiminto / tehtävä, laitteisto, toimipiste
 - toiminnan kohde, ajoitus, jne.
 - jako on yleensä monitasoinen, näkökulma voi vaihdella tasoittain ja olla eri osajärjestelmissä erilainen (vrt. "4+1 View Model")
- © Harri Laine, Jukka Paakki 15

- ### Kerrostus
- Suunnittelun ositusperiaatteet:
 - kerrostus (layering)
 - pilkkominen (partitioning)
 - Kerrostuksen perustana abstraktiotasot:
 - toiminta-abstraktiot
 - abstrakti kone -ajattelu
 - ylemmän kerroksen palvelut tuotetaan alemman kerroksen palveluiden avulla (vrt. tietoliikenteen OSI-malli)
 - kerrokset perustuvat eri käsitteistöihin
 - ylin kerros vastaa järjestelmän päätoimintoja ja käyttöliittymää, alin kerros usein fyysistä laitteistoa
- © Harri Laine, Jukka Paakki 16

- ### Kerrostus
- suljetussa kerrosarkkitehtuurissa vain välittömästi alemman tason palvelut käytettävissä
 - parempi ylläpidettävyys
 - avoimessa kerrosarkkitehtuurissa kaikkien alemmien tasojen palvelut käytettävissä
 - huonompi ylläpidettävyys
 - parempi joustavuus
 - esimerkki kerrostuksesta:
 3. Käyttöliittymäkerros
 2. Looginen toiminta -kerros
 1. Käyttöjärjestelmäkerros
 - laitteiston/järjestelmän eristäminen rajapinnan taakse
 - JDBC API kätkee tietokannan hallinnan
- © Harri Laine, Jukka Paakki 17



Kerrostus

- Tieto-abstraktiot
 - tietokantojen peruskäsitteitä
 - näkymät - looginen taso - fyysinen taso
 - abstraktit tietotyypit - oliot
 - tietokannan piilotus esim EJB-komponenttien taakse

© Harri Laine, Jukka Paakki 19

Piikkominen

- **Piikkominen** = toiminnallinen tai tiedollinen osiinjakko, ilmenee puhtaimmillaan asteittain tarkentuvassa suunnittelussa (top down)
 - asteittain voi tarkentaa sekä toimintaa että tietoa
 - esim. JSD, JSP (Jackson Structured Design, Jackson Structured Programming)

© Harri Laine, Jukka Paakki 20

Piikkominen

- Selvitä ongelman olennaiset osat.
- Hahmota ainakin yksi mahdollinen ratkaisu
 - mielellään useita ratkaisuja
 - simuloi keskeiset tilanteet
 - valitse yksinkertaisin ratkaisu, ellei ole erityisiä syitä valita toisin
- Kuvaa järjestelmä hierarkkisesti tarkentaen
 - ylhäältä alaspäin (suurempi kokonaisuus → pienempi kokonaisuus)
 - kuvausmenetelmä sovelluksen mukaan
- Muodosta vastaava rakennekaavio
- Kuvaa kunkin tarkennustason oliot

© Harri Laine, Jukka Paakki 21

Piikkominen: toiminta

© Harri Laine, Jukka Paakki 22

Piikkominen: data

© Harri Laine, Jukka Paakki 23

Piikkominen: toiminnallinen hierarkia

© Harri Laine, Jukka Paakki 24

Yleiset suunnitteluperiaatteet

- Suunnitteluprosessin pitää olla laajakatseista. Hyvä suunnittelija miettii myös vaihtoehtoisia toteutustapoja.
- Olemassaolevia komponentteja tulee käyttää hyväksi suunnittelussa.
- Suunnittelussa pitää mahdollisuuksien rajoissa matkia ratkaistavan ongelma-alueen rakennetta.
- Suunnitelman pitää olla yhtenäinen ja hyvin integroitu. Yhtenäisyys tarkoittaa, että suunnitelma näyttää yhden ihmisen tuotteelta. Integrointi tarkoittaa, että komponenttien rajapinnat on suunniteltu huolellisesti.

© Harri Laine, Jukka Paakki 25

Yleiset suunnitteluperiaatteet

- Suunnitelman pitää tukea mahdollisia muutoksia.
- Myös poikkeustilanteisiin pitää varautua. Mahdollisen toipumisen pitää olla käyttäjäystävällistä. Jos toipuminen ei ole mahdollista, ohjelmiston tulee päättyä selkeästi ja antaa informatiivinen virheilmoitus.
- Suunnittelu- ja toteutusvaiheet tulee erottaa selkeästi toisistaan. Suunnitelma on abstraktio toteutuksesta.
- Suunnitelman laadusta täytyy pitää kiinni jo suunnittelun aikana eikä pelkästään korjata suunnitelmaa lopussa.

© Harri Laine, Jukka Paakki 26

Yleiset suunnitteluperiaatteet

- Suunnitelmaa pitää arvioida suunnittelun aikana. Näin vähennetään suunnittelussa esiintyviä virheitä. Esim. formaalit tekniset tarkastukset sopivat erittäin hyvin tähän tarkoitukseen. Ylemmän abstraktiotason puutteet ja virheet pitää käsitellä ennen alemmalle abstraktiotasolle siirtymistä.
- Yksinkertaisuus on valttia jatkolle.

© Harri Laine, Jukka Paakki 27

Ohjelmarakenteita

spagettikoodi

ohjelmointikikkoja, data ja koodi jäsentymätöntä

© Harri Laine, Jukka Paakki 28

Ohjelmarakenteita

rakenteinen

asteittain tarkennettu tieto ja toiminta rakenteinen ohjelmointi

© Harri Laine, Jukka Paakki 29

Ohjelmarakenteita

oliokeskeinen

data ja koodi pakattuna rajapintojen taakse

© Harri Laine, Jukka Paakki 30

Ohjelmarakenteita

Oliokeskeinen

- modulaarisuus
- abstraktit tietotyypit
 - datan ja koodin pakkaaminen
- tiedon piilottaminen, rajapinnat
- olio-ohjelmointikielät
- olioperustaiset suunnittelumenetelmät (esim. UML ja Rational Unified Process (RUP))
- raviolikoodi?

© Harri Laine, Jukka Paakki 31

Hyvä ohjelmistorakenne

Yleisiä ominaisuuksia, joihin on syytä kiinnittää huomiota:

- modulaarisuus (modularity)
- kiinteys (cohesion)
- kytkentä (coupling)
- tiedon kätkeminen (information hiding)
- muunneltavuus (modifiability)
- ylläpidettävyys (maintainability)

© Harri Laine, Jukka Paakki 32

Hyvä ohjelmistorakenne – modulaarisuus

- Modulaarisuus: ohjelmisto on jaettu erillisiin nimettyihin (usein erikseen käännettäviin) osiin eli moduuleihin
 - tietorakenteita
 - yleensä yksityisiä, käyttö operaatioiden kautta
 - moduuli voi tarjota myös jaetun tietorakenteen
 - operaatioita
 - julkisia tai yksityisiä
 - liittymä
 - julkiset operaatiot, joita muut moduulit voivat kutsua

© Harri Laine, Jukka Paakki 33

Hyvä ohjelmistorakenne – modulaarisuus

Modulaarisuus:

$Effort(p+q) > Effort(p) + Effort(q)$

moduulin koko

- Moduulien määrän kasvaessa liittymien toteutuskustannukset kasvavat, joten jatkuva pilkkominen pienemmiksi moduuleiksi ei poista kaikkia kustannuksia
- Suositeltu moduulikoko noin 100-200 riviä koodia, yksittäiset operaatiot 20-30 riviä
 - kerralla hahmotettavissa

© Harri Laine, Jukka Paakki 34

Hyvä ohjelmistorakenne – kiinteys

- Kiinteys (cohesion) (toiminnallinen)
 - moduuli on kiinteä, jos sen kaikki osat palvelevat yhtä ainoaa tehtävää, ts. moduuli toteuttaa yhden selkeän toiminnallisen kokonaisuuden
 - moduuli ei ole kiinteä, jos se sisältää heikosti toisiinsa liittyviä osia
 - kiinteys parantaa moduulin ylläpidettävyttä ja lisää mahdollisuuksia uskäyttöön

© Harri Laine, Jukka Paakki 35

Hyvä ohjelmistorakenne – kiinteys

Kiinteyden asteet huonosta hyvään:

- moduulissa on satunnaisesti yhdistettyjä osia
- moduuliin on koottu loogisesti samantyyppiset tehtävät (esim. tulostukset)
- moduuliin on koottu samaan aikaan tarvittavat osat (esim. alustukset)
- moduuliin on koottu kontrollin etenemisjärjestyksessä peräkkäisiä osia
- moduuliin on koottu yhteisiä tietorakenteita käyttäviä operaatioita (olio)
- moduulin osat muokkaavat toistensa tuloksia: toisen tuote on toisen syöte
- toimintakokonaisuuteen perustuva yhdistäminen: jokainen osa omalta osaltaan tarpeen moduulin vastuulla olevan toimintakokonaisuuden kannalta

© Harri Laine, Jukka Paakki 36

Hyvä ohjelmistorakenne – kytkentä

- **Kytcentä (coupling)**
 - kuvaa vuorovaikutuksia eri moduulien välillä
 - **löyhästi kytkettyjä** moduuleja on helpompi ylläpitää, koska muutokset eivät vaikuta muihin moduuleihin
 - kiinteästi kytketyillä on voimakkaita riippuvuuksia
 - "kytkös": yleensä moduulin jonkin operaation / metodin kutsu
 - muita tyypillisiä kytköksiä: perintäsuhde, määrittelysuhde

© Harri Laine, Jukka Paakki 37

Hyvä ohjelmistorakenne – kytkentä

Kytcentä löyhästä vahvaan:

- ei vuorovaikutusta moduulien välillä
- käsiteltävän tiedon välittäminen yksittäisinä parametreina
- käsiteltävän tiedon välittäminen rakenteisina parametreina
- ohjaustiedon välittäminen parametreina
- yhteiskäyttöiset ulkoiset laitteet
- yhteiskäyttöinen tietorakenne
- toisten moduulien sisäisten rakenteiden hyväksikäyttö

© Harri Laine, Jukka Paakki 38

Hyvä ohjelmistorakenne – tiedon piilotus

- **Tiedon piilotus (information hiding)**
 - moduulia ajatellaan mustana laatikkona
 - vain liittymät näkyvät ulospäin :
 - syötteet
 - tulosteet
 - moduulin sisältämien toimintojen määrittely (kutsurajapinta)
 - ulos eivät näy, eivät ulkopuolelta käytettävissä:
 - moduulin sisäiset tietorakenteet
 - toimintojen toteutustapa
 - toteutus, testaus ja ylläpito paikallista
 - suojaus väärinkäyttöä vastaan tehostuu

© Harri Laine, Jukka Paakki 39

Hyvä ohjelmistorakenne

- Muita jaossa vaikuttavia asioita:
 - jakoa ei kannata jatkaa, jos liittymän koodi tulee pidemmäksi kuin toiminnan koodi (**paitsi yleiskäyttöisten osalta**)
 - yhteiskäyttöisten osien eristäminen omiksi moduuleikseen lyhentää koodia ja helpottaa ylläpitoa
 - työ ja päätöksenteko kannattaa erottaa eri moduuleihin, muutokset kohdistuvat yleensä vain jompaan kumpaan
 - tässä yhteydessä on kuitenkin pyrittävä välttämään toimivaltariikkomuksia = tilanteita, joissa päätöksen vaikutusalue menee moduulin valvonta-alueen (= alaiset kutsurakenteessa) ulkopuolelle
 - kannattaa etsiä yleiskäyttöisiä moduuleita

© Harri Laine, Jukka Paakki 40

Hyvä ohjelmistorakenne

- Muita jaossa vaikuttavia asioita:
 - liittymien pitäisi olla mahdollisimman yksinkertaisia, ei rakenteista tietoa
 - moduulirakenteeseen syvyyttä leveyden kustannuksella
 - kuitenkin: **luokkahierarkiassa** syvää perintää vältettävä, koska se häiltää ymmärrettävyyttä ja ylläpidettävyyttä
 - moduulin toiminnan tulee olla ennustettavissa – ts. sisäinen tila ei saisi vaikuttaa (black box)
 - moduuleilla pitäisi olla vain yksi sisäänmeno- ja yksi ulostulokohta (esim. määritelty kutsurajapinta)
 - mittareita: **fan_in** (moduulia kutsuvien moduulien määrä), **fan_out** (moduulin kutsuvien moduulien määrä), moduulihierarkian **maksimisyvyys** ja **maksimileveys**

© Harri Laine, Jukka Paakki 41

Moduulihierarkia – kutsuriippuvuus

© Harri Laine, Jukka Paakki 42

Moduulihierarkia – kutsuriippuvuus

- **Moduulihierarkian suunnittelu:**
 - relaatio **A uses B** on voimassa, mikäli moduulille A määritelty tehtävä vaatii moduulin B käyttämistä (kutsumista)
 - relaation **uses** tuottama riippuvuusverkko on kehätön =>
 - verkko muodostaa moduulien (puumaisen) hierarkian
 - tavoitteena on kehätön verkko
 - esim: a **uses** b, b **uses** c, a **uses** c, d **uses** c

© Harri Laine, Jukka Paakki 43

Moduulihierarkia – kutsuriippuvuus

- nyt jokainen taso koostuu moduulien osajoukosta, joka voidaan **toteuttaa, suorittaa ja testata** erikseen **inkrementaalisesti**

testausjärjestys integroititestauksessa

© Harri Laine, Jukka Paakki 44