

Arkkitehtuurityylejä ja patterneja

Luento 4

10.9.2015

581358 Ohjelmistoarkkitehtuurit

1

Oppimistavoitteet

- Esimerkkejä yleisesti käytetyistä arkkitehtuuripattereista ja -tyyleistä
- Muutama käydään läpi perusteellisemmin (MVC, tietovuotyylit, kerrosarkkitehtuurit)

10.9.2015

581358 Ohjelmistoarkkitehtuurit

2

Tyylien ja patternien käytöstä

- Platoninen vs. "ruumiillistunut" tyyli/patterni
 - Platoninen tyyli/patterni on ideaalinen kuvaus arkkitehtuuriratkaisusta
 - Käytännön toteutuksissa esiintyvät tyylien/patternien instanssit ("ruumiillistumat") ovat harvoin täysin "puhtaita", eli niissä on tehty jotain perusteltuja poikkeuksia ja lisäksiä ratkaisun rakenteisiin ja rajoitteisiin
 - Tyylit ja patternit ovat yleistyksiä
 - Yleistys, joka määrittelee arkkitehtuurien *luokan*
 - On myös paljon sovellusalue- ja teknologiakohtaisia patterneja, joilla on rajallisempi sovellusala

10.9.2015

581358 Ohjelmistoarkkitehtuurit

3

Esimerkkejä

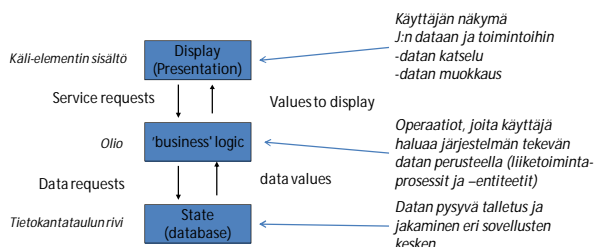
ARKKITEHTUURIPATTERNEJA

10.9.2015

581358 Ohjelmistoarkkitehtuurit

4

Kolmitasoarkkitehtuuri (N-Tier)



10.9.2015

581358 Ohjelmistoarkkitehtuurit

5

Kolmitasoarkkitehtuuri (N-Tier)

- Sovellusalue
 - Hajautetut, data-intensiiviset informaatiojärjestelmät
- Edut
 - Helpottaa tietoturvan, suorituskyvyn ja saatavuuden optimointia eri tasoilla niille sopivilla tavoilla
 - Lisää järjestelmän modulaarisuutta ja muunneltavuutta, koska eri tasojen välille täytyy sopia määrämuotoiset kommunikaatioprotokollat (-> loose coupling)
- Haitat
 - Kustannukset, monimutkaisuus

10.9.2015

581358 Ohjelmistoarkkitehtuurit

6

Model-View-Controller (MVC) [malli-näkymä-ohjain]

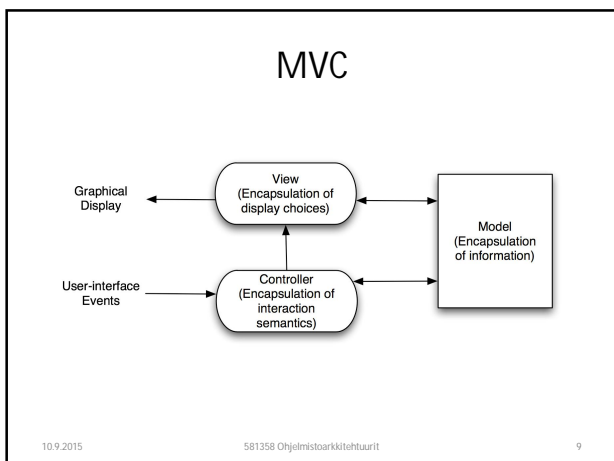
- Sovellusalue: monimuotoisia käyttöliittymänäkymiä sisältävät interaktiiviset järjestelmät
- Motivaatio:
 - Ohjelmistolla on erilaisia käyttäjiä ja näillä erilaisia tarpeita käyttöliittymään liittyen
 - Samaa informaatiota pitää pystyä esittämään ja käsittelemään eri muodoissa
 - Käyttöliittymän tulisi olla helposti muutettavissa
- Krasner, G., Pope S: Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80, *JOP* Aug./Sep., 1988.

10.9.2015 581358 Ohjelmistoarkkitehtuurit 7

MVC

- Kolmenlaisia komponentteja
 - Malli-komponentit ovat vastuussa laskentaan liittyvän tiedon (tai tilan) säilytyksestä
 - Näkymä-komponentit ovat vastuussa tiedon esittämisestä käyttäjälle
 - Ohjain-komponentit ovat vastuussa käyttäjän ja järjestelmän välisen vuorovaikutuksen ohjaamisesta
 - Ottavat esimerkiksi vastaan hiiren näpäyksiä, näppäimen painamisia jne. ja muuntavat nämä mallikomponenttien tilaan vaikuttaviksi operaatioiksi

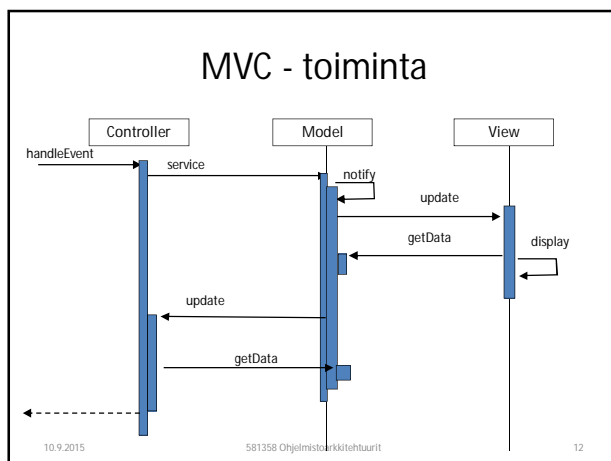
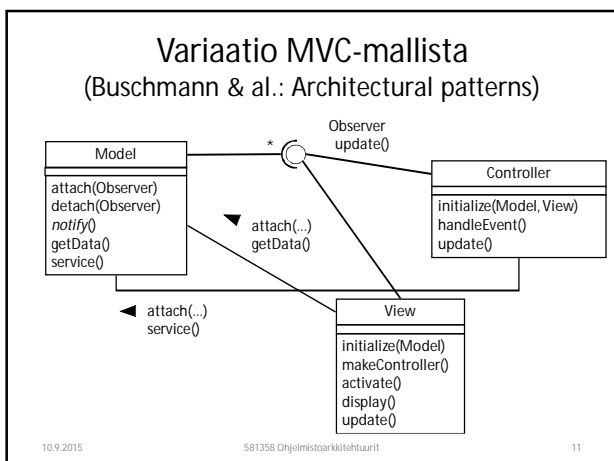
10.9.2015 581358 Ohjelmistoarkkitehtuurit 8



MVC

- Mallikomponentit eivät riipu mistään (tietystä) näkymä- tai ohjainkomponentista
- Muutokset mallikomponenteissa välitetään näkymille *Tarkkailija*-suunnittelumallin mukaisesti
 - Eli vain niille näkymille, jotka ovat ilmaisseet kiinnostuksensa muutoksiin

10.9.2015 581358 Ohjelmistoarkkitehtuurit 10



MVC - arviointia

- Etuja
 - Useita näkymiä samaan tietoon
 - Näkymiä on helposti lisättävissä jopa dynaamisesti
 - Ohjaimen voi helposti vaihtaa
 - Voi toimia perustana sovelluskehyselle (Smalltalk, Web - applikaatiot)
- Ongelmia
 - Lisää mutkikkuutta
 - Yksinkertainen käyttäjätoimenpide saattaa aiheuttaa monia päivityksiä näkymään (päivityksen laajuutta voi olla vaikea kontrolloida)
 - Voi olla työlästä löytää sopiva datan esitysmuoto ja haku-/päivitysoperaatiot mallin rajapintaan käyttöliittymän (näkyvän) suorituskyvyn (responsiivisuus) kannalta

10.9.2015

581358 Ohjelmistoarkkitehtuurit

13

Muunnelmia

- Monesti ei ole tarpeen erottaa *ohjainta* ja *näkymää*
 - Erityisesti, jos kullakin näkymällä olisi joka tapauksessa oma ohjain, jonka kautta voidaan suoraan manipuloida näkymää
 - Käytännössä monissa käyttöliittymäkehysissä (GUI frameworks) "kontrollerikoodi" liitetään UI elementteihin suoraan tapahtumankäsittelijöinä
- Ts. komponentti voi toimia sekä näkymä- että ohjainkomponenttina. Yhteen malliin voi liittyä monta <näkymä, ohjain> -paria.

10.9.2015

581358 Ohjelmistoarkkitehtuurit

14

Web-MVC

- MVC –patternin idea erottaa sovelluksen data ja tila näiden esittämisestä käyttäjälle on niin hyödyllinen, että sitä sovelletaan laajasti web-sovellusten toteutuksessa
 - Myös eri mittakaavoissa, eli sekä web-palvelimen että asiakkaan (selain) puolella
- MVC:n käyttö palvelinpuolella: <http://advancedkittenry.github.io/koodaaminen/arkkitehtuuri/index.html>

10.9.2015

581358 Ohjelmistoarkkitehtuurit

15

Esimerkkejä

ARKKITEHTUURITYYLIT

10.9.2015

581358 Ohjelmistoarkkitehtuurit

16

Tyylien luokittelua

- Eri lähteet luokittelevat tyyliä eri tavoin
 - Fairbanks ei esimerkiksi luokittele tyyliä lainkaan
 - Bass, Clements ja Kazman (*Software Architecture in Practice*, 3. painos) taas kutsuvat tyyliä patterneiksi ja luokittelevat ne omalla tavallaan
- Seuraavassa noudatetaan kurssin edellisen oppikirjan (Taylor & al.) luokitusta (ei kattava - luokat eivät pistevieraita)
- Tarkastellaan muutamaa esimerkkiä eri luokista
 - Tietovuopohjaiset tyylit
 - Kerrosrakenteiset tyylit

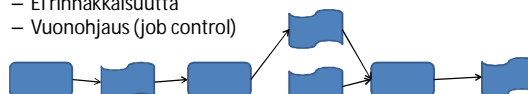
10.9.2015

581358 Ohjelmistoarkkitehtuurit

17

Tietovuopohjaiset tyylit

- *Data flow styles*
- Pääpaino siinä miten tieto liikkuu riippumattomien komponenttien välillä
- *Erätöiden sarja* (batch sequential)
 - Perinteinen eräajosovellus
 - Tiedon käsittely jakautuu vaiheisiin,
 - Lopputulos valmis kun kaikki vaiheet on suoritettu
 - Ei rinnakkaisuutta
 - Vuonohjaus (job control)



10.9.2015

581358 Ohjelmistoarkkitehtuurit

18

Tietovuopohjaiset tyylit

- Erätöiden sarja jatkuu:
 - Tieto liikkuu kokonaisuuksina vaiheelta toiselle esimerkiksi ajastetusti, vuonohjauskielen tai käyttäjän ohjaamana
 - Vaiheelle voi tulla useita syöttöaineistoja ja se voi tuottaa useita tulosaineistoja
 - Vaihe käsittelee syöttöaineistot kokonaisuudessaan ennen tulosaineiston luovutusta eteenpäin seuraavalle vaiheelle
 - Sovellusalueita:
 - Monivaiheiset tiedonjalostusprosessit, jossa aineistoa ei voi käsitellä osina
 - XML-muunnokset käyttäen dokumenttipuuta

10.9.2015

581358 Ohjelmistoarkkitehtuurit

19

Tietovuopohjaiset tyylit

- Erätöiden sarja - etuja:
 - Muunneltavuus, vaiheet ovat riippumattomia toisistaan (vunohjaus hoitaa koordinoinnin)
 - Yksinkertainen rakenne, suoritukset etenee yksi vaihe kerrallaan
 - Potentiaalisesti hyvä suorituskyky (throughput)
- Haittoja:
 - Tulos on valmis ja saatavilla vasta, kun viimeinen vaihe on valmis (ei inkrementaalisuutta)
 - Rinnakkaisuutta ei voi hyödyntää

10.9.2015

581358 Ohjelmistoarkkitehtuurit

20

Tietovuopohjaiset tyylit

- Väylät ja suodattimet (pipes and filters)
 - Koostuu *prosessointiyksiköistä* (filters) ja niitä yhdistävistä tietovirtaa kuljettavista *väylistä* (pipes)
 - Prosessointiyksiköt käsittelevät aktiivisesti tietoa
 - Väylät siirtävät tietoa eteenpäin passiivisesti
 - Prosessointiyksiköt ovat ("puhtaassa tapauksessa") täysin itsenäisiä
 - Lukevat syötevirtaa ja tekevät sen perusteella laskentaa → lähettävät eteenpäin muokatun syötevirran

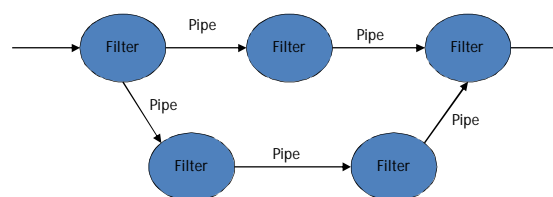
10.9.2015

581358 Ohjelmistoarkkitehtuurit

21

Tietovuopohjaiset tyylit

Väylät ja suodattimet jatkuu...



10.9.2015

581358 Ohjelmistoarkkitehtuurit

22

Tietovuopohjaiset tyylit

- Väylät ja suodattimet jatkuu...
- Kukin prosessointiyksikkö pitää voida toteuttaa itsenäisenä entiteettinä
 - Ei jaettava tilatietoa
 - Prosessointiyksikkö riippuu ainoastaan oman syötteensä muodosta
- Prosessointi tapahtuu yhdessä vaiheessa pala kerrallaan:
 - tietoalkion (esimerkiksi tekstirivin) prosessointi ei saisi riippua jonkin tulevan tietoalkion prosessoinnista
 - tällöin suodattimet voivat toimia rinnakkain
 - Unix-väyliin kytketyt käsittelyvaiheet ovat usein sellaisia, että rinnakkaisuus estyy – käsittely voi edetä vasta kun koko aineisto saatu, esim järjestäminen suodattimena
 - Tietoa voidaan puskuroida, mutta laajamittainen puskurointi rikkoo arkkitehtuurin perusajatuksen, ja vaikeuttaa esimerkiksi syöteen rinnakkaisista prosessointia

10.9.2015

581358 Ohjelmistoarkkitehtuurit

23

Tietovuopohjaiset tyylit

- Väylät ja suodattimet – etuja
 - Muunneltavuus, suodattimet itsenäisiä prosessoreita, järjestelmän helppo konfigurointi eri käyttötarkoituksiin (-> yksi Unix-komentorivi)
 - Rinnakkaisuuden hyödyntäminen
 - Tulosten inkrementaalinen saatavuus
- Haittoja
 - Ei sovi interaktiivisiin sovelluksiin
 - Koviin eri tahtiin toimivat suodattimet voivat aiheuttaa vaikeuksia väylien toteutukselle (puskurointi)
 - Syöteen loppumisen havaitseminen voi vaatia erikoistoimia

10.9.2015

581358 Ohjelmistoarkkitehtuurit

24

Tietovuopohjaiset tyylit

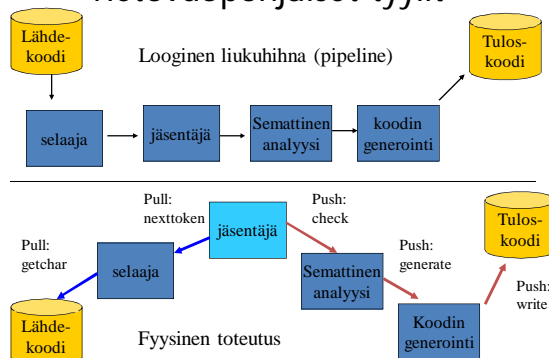
- Väylät ja suodattimet jatkuu...
- *Liukuhihna* (pipeline architecture)
 - Erikoistapaus väylät ja suodattimet mallista: Ei haarautumisia
 - Helppo synkronoida, voi edetä
 - työntämällä (push)
 - prosessointiyksikkö kutsuu seuraavan yksikön palvelua ja välittää tietoa eteenpäin (alkaa alusta)
 - tai vetämällä (pull)
 - prosessointiyksikkö kutsuu edellisen yksikön palvelua (rekursiivisesti).

10.9.2015

581358 Ohjelmistoarkkitehtuurit

25

Tietovuopohjaiset tyylit



10.9.2015

581358 Ohjelmistoarkkitehtuurit

26

Kerrosarkkitehtuurit

- *Layered architectures*
- Kerrosarkkitehtuuri koostuu tasoista, jota on järjestetty jonkin abstrahointiperiaatteen mukaan nousevaan järjestykseen
 - Usein abstrahointi tapahtuu akselilla ihminen-laite
 - Sovellus – Palvelu - Resurssi
 - Usein kerrokset ovat luonteeltaan niin teknisiä, että eri abstraktiotasojen nimeäminen ja erottaminen käsitteellisesti toisistaan on hankalaa
 - Alemman tason palvelut ovat tyypillisesti primitiivisempiä ja yleiskäyttöisempiä kuin ylemmän tason palvelut

10.9.2015

581358 Ohjelmistoarkkitehtuurit

27

Kerrosarkkitehtuurit

- Kerrokset ovat yleensä nähtävissä yhden suoritusympäristön (tietokoneen, solmun) sisällä suoritettavan ohjelmiston rakennetta ja sen käyttämiä palveluja tarkasteltaessa
 - Eri kerrokset eivät yleensä ole hajautettuja eri solmuihin kuten kolmitasoarkkitehtuurissa

10.9.2015

581358 Ohjelmistoarkkitehtuurit

28

Kerrosarkkitehtuurit

- Virtuaalikoneet (virtual machines) kerrosarkkitehtuuriyppinä
- Perusajatus:
 - Taso toimii virtuaalikoneena, joka tarjoaa palveluita ylemmän tason korkeamman abstraktion virtuaalikoneelle. Ylemmän tason tarjoamat palvelut toteutetaan välittömästi alemman tason palveluita hyväksikäyttäen.
 - Ajatus toteutuu vain *puhtaassa* (strict) kerrosarkkitehtuurissa

10.9.2015

581358 Ohjelmistoarkkitehtuurit

29

Kerrosarkkitehtuurit

- Käytännössä puhtaasta kerrosarkkitehtuurista voi olla kahdenlaisia poikkeamia:
 - Palvelukutsuja tehdään alemmasta kerroksesta ylemmän kerrokseen
 - Palvelukutsu ohittaa kerroksia kulkiessaan ylhäältä alaspäin (avoin kerrosarkkitehtuuri)
- Kerrosten ohittaminen on tarpeen useimmiten tehokkuussyistä (tai sitten välissä oleva kerros ei toteuta tiettyä palvelua, joka löytyy alemmalta kerrokselta)

10.9.2015

581358 Ohjelmistoarkkitehtuurit

30

Kerrosarkkitehtuurit

- Kerroksen ohittamisesta aiheutuva ongelma:
 - Tietty kerros tulee riippuvaiseksi myös muista kuin suoraan sen alapuolella olevasta kerroksesta
 - Kerroksen vaihtaminen hankalaa
- Palvelukutsun tekeminen alemmasta kerroksesta ylemmään päin voi olla merkki vakavasta ongelmasta arkkitehtuurissa
 - Alempi kerros saattaa olla riippuvainen ylemmästä
 - Syklinen riippuvuus

10.9.2015

581358 Ohjelmistoarkkitehtuurit

31

Kerrosarkkitehtuurit

- Joskus alemman kerroksen on kuitenkin tarpeen kutsua ylemmän kerroksen koodia
 - Alemman kerroksen täytyy mukauttaa toimintaansa ylemmän kerroksen mukaan
 - Jotta alempi kerros ei tulisi (liian) riippuvaiseksi ylemmästä kerroksesta, voidaan hyödyntää *takaisinkutsuperiaatetta* (callback)
 - Alempi kerros tarjoaa rekisteröintioperaation, jonka avulla ylempi kerros rekisteröi takaisinkutsussa kutsuttavan koodin (funktion) alemman kerroksen käyttöön
 - Alempaa kerrosta ei kiinnosta, mitä ylemmän kerroksen takaisinkutsufunktio tekee, se vain kutsuu sitä tietyssä tilanteessa

10.9.2015

581358 Ohjelmistoarkkitehtuurit

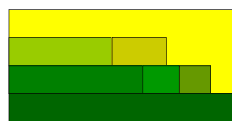
32

Kerrosarkkitehtuurit

Ei ohiteta kerroksia:



Ohitetaan kerroksia:



10.9.2015

581358 Ohjelmistoarkkitehtuurit

33

Kerrosarkkitehtuurit

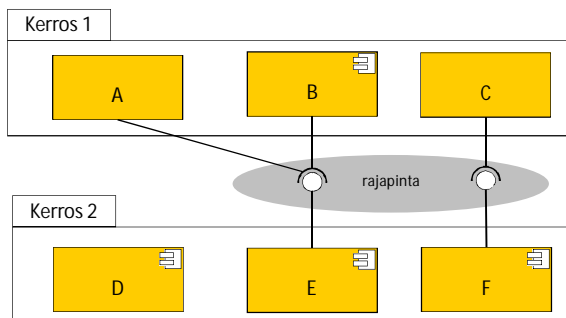
- Kun kerrosten väliset rajapinnat on selkeästi määritellyt, kerros voidaan vaihtaa toiseksi ilman, että se vaikuttaa muuhun järjestelmään
- rajapintojen/ kerrosten toimintojen *standardointi* lisää vaihdettavuutta

10.9.2015

581358 Ohjelmistoarkkitehtuurit

34

Kerrosarkkitehtuurit

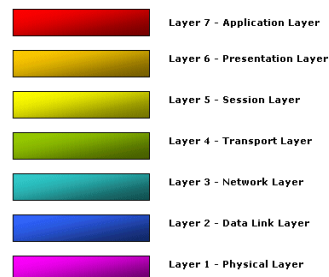


10.9.2015

581358 Ohjelmistoarkkitehtuurit

35

Kerrosarkkitehtuurit – OSI¹



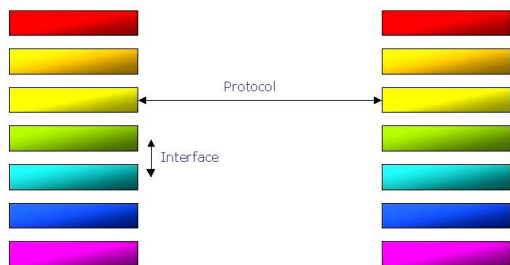
Eräänlainen referenssiarkkitehtuuri kommunikaatiojärjestelmille
¹http://en.wikipedia.org/wiki/OSI_model

10.9.2015

581358 Ohjelmistoarkkitehtuurit

36

Kerrosarkkitehtuurit - OSI



Vastinkerrokset kommunikoivat samalla abstraktiotasolla (yhteisellä protokollalla)

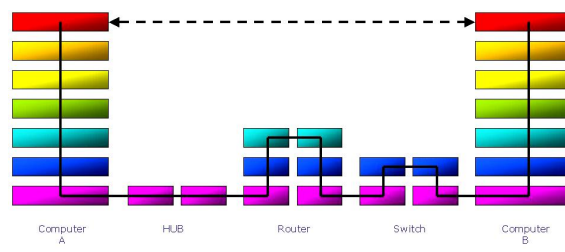
10.9.2015

581358 Ohjelmistoarkkitehtuurit

37

Kerrosarkkitehtuurit - OSI

Kokonaiskuva

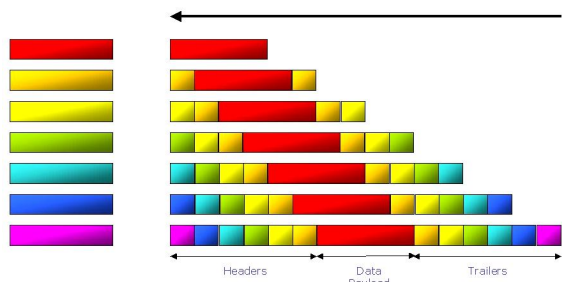


10.9.2015

581358 Ohjelmistoarkkitehtuurit

38

Kerrosarkkitehtuurit - OSI



Jokainen kerros lisää oman palansa viestiin sen kulkiessa kerrosten läpi

10.9.2015

581358 Ohjelmistoarkkitehtuurit

39

Kerrosarkkitehtuurit – muita esimerkkejä

- Virtuaalikoneella rajapinta tai kieli (esim. Java bytecode), jonka avulla konetta käytetään
 - kätkee toteutuksen, mahdollistaa useita alustoja
- Sovellusrajapinnat palveluihin (API, application programming interface)
 - Esimerkiksi käyttöjärjestelmillä, luokkakirjastoilla → kätkee yksityiskohdat, parantaa siirrettävyyttä
- Asiakas-palvelin arkkitehtuuri
 - 2-tasoinen kerrosarkkitehtuuri
 - etäproseduurikutsut kommunikoinnissa

10.9.2015

581358 Ohjelmistoarkkitehtuurit

40

Kerrosarkkitehtuurit

- Etuja:
 - Kerrosarkkitehtuuri on yleinen malli, jota voidaan soveltaa lähes kaikissa järjestelmissä pienessä tai isossa mittakaavassa
 - Jakaa järjestelmän korkealla tasolla osiin, joiden muodostamana kokonaisuutena järjestelmä on helppo ymmärtää
 - Intuitiivinen, helposti ymmärrettävä → voidaan käyttää kommunikointiin hyvin erilaisten sidosryhmien välillä
 - Ohjaa ohjelmiston riippuvuuksien minimointiin → muutosten paikallisuus → helppo ylläpidettävyyys, muutettavuus, vaihdettavuus, *standardoitavuus*
 - Kerrosarkkitehtuuri tukee ohjelmiston uudelleenkäyttöä
 - Kukin kerros toteuttaa tietyt abstraktion → pohja työn jakamiselle → tasokohtaista erikoistumista

10.9.2015

581358 Ohjelmistoarkkitehtuurit

41

Kerrosarkkitehtuurit

- Ongelmia:
 - Tehokkuushäviö
 - Palvelukutsu ei välity suoraan palvelun toteuttavalle kerrokselle, vaan kulkee välissä olevien kerrosten kautta
 - Palvelukutsun parametrit mahdollisesti muutetaan joka kerroksen välillä uuteen esitysmuotoon
 - Toteutustaakka: periaatteessa jokainen palvelu on toteutettava joka tasolla (vaikkakin osa toteutuksesta delegoidaan alemmalle kerrokselle)
 - Oikean (toimivan) tasojaon keksiminen on hankalaa
 - Mihin kerrokseen tietty palvelu pitäisi liittää?

10.9.2015

581358 Ohjelmistoarkkitehtuurit

42

Kerrosarkkitehtuurit

- Ongelmia:
 - *Poikkeusten käsittely*
 - Kutsu ylemmältä kerrokselta →
 - poikkeus kerrosrakenteen pohjalla →
 - palataan kutsupinossa taaksepäin kunnes löytyy käsittelijä poikkeukselle →
 - poikkeuksen käsittely ylemmällä tasolla kuin missä poikkeus tapahtui →
 - käsittelijä ei välttämättä pysty korjaamaan tilannetta
 - Ääritapaus: poikkeus palaa käyttäjälle saakka antaen mystisen virheilmoituksen
 - Miksei sitten poikkeuksia käsitellä aina siellä missä ne syntyvät?