

Ohjelmistoarkkitehtuurin suunnittelu

Luento 3

16.9.2016

581358 Ohjelmistoarkkitehtuurit

1

Oppimistavoitteet

- Arkkitehtuuritietämyksen lähteet
- Arkkitehtuurityyli ja -patternit
- Yleisiä suunnitteluperiaatteita
- Ositusstrategiat
- "Kaunis" arkkitehtuuri

16.9.2016

581358 Ohjelmistoarkkitehtuurit

2

ARKKITEHTUURITIEDON LÄHTEET

16.9.2016

581358 Ohjelmistoarkkitehtuurit

3

Ohjelmistoarkkitehtuuritiedon lähteillä

- Yhdellä (yliopisto-) kurssilla ei kenestäkään kouluteta ohjelmistoarkkitehtia
- Ohjelmistoarkkitehdiksi *kasvetaan* kokemuksen ja haastavien tehtävien kautta
- Muitten kokemuksesta voi kuitenkin ottaa oppia ja kehittää tietojaan ja suunnittelutaitojaan
 - Arkkitehtuurityylit ja -patternit
 - Yleiset ohjelmistojen suunnitteluperiaatteet
 - Laatuominaisuuksien suunnittelutaktiikat
 - Kokemusraportit ja kuvaukset onnistuneista ja epäonnistuneista kehitysprojekteista (blogit, kirjat)

16.9.2016

581358 Ohjelmistoarkkitehtuurit

4

Referenssejä

M. Fowler



Pearson, 2003

<http://martinfowler.com/eaCatalog/>
L. Bass, P. Clements,
R. Kazman:

Addison-Wesley Prof., 2012

I. Mistrik, A. W. Brown,
M. Ali BabarElsevier /Morgan Kaufmann,
2013

D. Spinellis, G. Gousios



O'Reilly, 2009

16.9.2016

581358 Ohjelmistoarkkitehtuurit

5

Lisää referenssejä

- High Scalability –sivusto
<http://highscalability.com/>
 - "Real Life Architectures"
 - Erityisesti web-sovellusten ja palvelujen rakentajille
- Handbook of Software Architecture (status??)
<http://handbookofsoftwarearchitecture.com>
 - Katso: *Books*

16.9.2016

581358 Ohjelmistoarkkitehtuurit

6

ARKITEHTUURITYYLIT JA - PATTERNIT

16.9.2016

581358 Ohjelmistoarkkitehtuurit

7

Arkkitehtuurityyli

- *Architectural style*
 - Nimetty kokoelma tiettyyn käyttöyhteyteen soveltuvia yleisiä suunnitteluperiaatteita ja sääntöjä, jotka tuovat mukanaan hyödyllisiä ominaisuuksia rakennettavaan järjestelmään
- Esim. *asiakas – palvelin* tyyli (Client-Server):
 - Erottele palvelua **pyytävä** ja palvelun **tarjoava** ohjelmistokomponentti
 - Piilota palvelua pyytävien komponenttien identiteetti palvelun tarjoajalta ja mahdollista useiden pyytäjien mahdollisesti vaihtelevan joukon palveleminen
 - Eristä pyytäjät toisistaan
 - Mahdollista useita palvelun tarjoajia; ja mahdollista tarjoajien määrän dynaaminen (käytön aikainen) lisääminen asiakkaiden määrän vaihtelun mukaan

16.9.2016

581358 Ohjelmistoarkkitehtuurit

8

Arkkitehtuuripatterni (tai –malli)

- *Architectural pattern*
 - Nimetty kokoelma johonkin toistuvaan suunnitteluongelmaan soveltuvia suunnitteluratkaisuja, jotka on parametrisoitu ottamaan huomioon käyttöyhteys, jossa ongelma esiintyy
- Miten tyyli eroaa patternista?
 - Tyylin käyttötilanne on yleisempi, patternin spesifimpi
 - Tyylit ovat enemmän periaatesääntöjä ja patternit konkreettisia ratkaisuja
 - Patternit soveltavat tyyliä (tyylejä)
 - Patterniin voidaan yhdistää tyylejä. Tietty tyyliä noudattava ratkaisu voi sisältää useita patterneja.
 - Huom - Kaikki lähteet eivät erottele näitä käsitteitä!

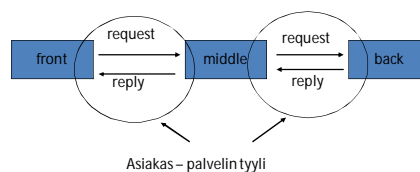
16.9.2016

581358 Ohjelmistoarkkitehtuurit

9

Tyyli ja patterni

- Esimerkki:
 - Kolmitasoarkkitehtuuri –patterni

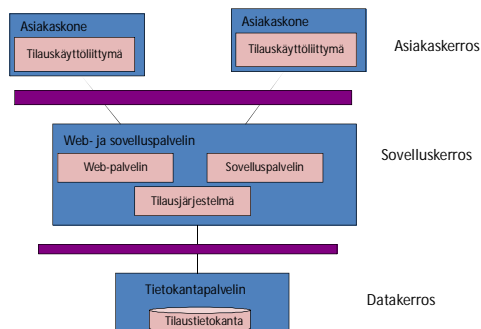


16.9.2016

581358 Ohjelmistoarkkitehtuurit

10

Kolmitasoarkkitehtuuri



16.9.2016

581358 Ohjelmistoarkkitehtuurit

11

Tyylien käytöstä

- Muitten jäljittely ja suunnittelun uudelleenkäyttö on hyvä oppimismenetelmä
- Kun käyttää tyylejä ja ratkaisumalleja, on kuitenkin syytä ymmärtää *miksi ja missä tilanteissa* ne toimivat ja verrata tätä omaan käsillä olevaan suunnitteluongelmaan
- Joukko tyylejä ja malleja käydään läpi seuraavilla luennoilla

16.9.2016

581358 Ohjelmistoarkkitehtuurit

12

YLEISET SUUNNITTELUPERIAATTEET

16.9.2016

581358 Ohjelmistoarkkitehtuurit

13

Yleiset suunnitteluperiaatteet

- *Abstraction, Encapsulation, Information Hiding, Modularization, Separation of Concerns, Coupling and Cohesion, Sufficiency-Completeness-Primitiveness, Separation of Policy and Implementation, Single Point of Reference, Divide-and-Conquer*
- Katso esimerkiksi
 - Luku 6.3 *Enabling techniques for software architecture* teoksessa Buschmann F. & al.: *Pattern-Oriented Software Architecture*, vol. 1. Wiley, 2001
 - Wikipedia
http://en.wikipedia.org/wiki/List_of_software_development_philosophies

16.9.2016

581358 Ohjelmistoarkkitehtuurit

14

Information Hiding

- Parnas, D.: *On the Criteria to Be Used in Decomposing Systems Into Modules*. Comm. ACM 15(12), 1972
 - Ohjelmisto jaetaan moduuleihin* siten, että kukin moduuli *kätkee* jonkin todennäköisesti *muuttuvan* teknisen tai sovellusalueen piirteen toteutuksen (= suunnittelupäätökset)
 - Moduuli tarjoaa palveluihinsa *vakaan abstraktin rajapinnan*, joka ei paljasta toteutuksen yksityiskohtia (-abstrakti tietotyypit)

*Moduuli (hist.) – ohjelman toteutuksen osa, joka sisältää yhteen kuuluvia elementtejä, esim. Java-pakkaus ja sen luokat. Staattinen rakenneosa (koodia, konf. tiedostoja tms.).

16.9.2016

581358 Ohjelmistoarkkitehtuurit

15

Information Hiding

- Kun moduulin kätkemät suunnittelupäätökset muuttuvat, moduulin käyttäjä ei tarvitse muuttaa, koska ne riippuvat vain samana pysyvistä rajapinnasta
- Useimmat ohjelmistokehittäjät pitävät itsestäänselvytenä, että rajapinta (esim. Java-olion) ei paljasta olion *toteutuksen* yksityiskohtia
 - Kentät merkitään yksityisiksi (private), mutta määritellään niille julkiset `get()` ja `set()` –metodit
- Harvempi kuitenkin tulee ajatelleeksi *odotettavissa olevia muutoksia* ja muutosten heijastusvaikutusten *ennalta ehkäisemistä* ohjelmiston modularisoinnin avulla
 - Arkkitehdin työn kuvaan tällaisen ajattelemisen kuitenkin kuuluu
- Muutostarpeita voi tosin olla vaikea arvata etukäteen
 - Tähän ongelmaan tepsii, paitsi oikein tehty olioperustainen suunnittelu, seuraavalla dialla esitelty periaate

16.9.2016

581358 Ohjelmistoarkkitehtuurit

16

Separation of Concerns

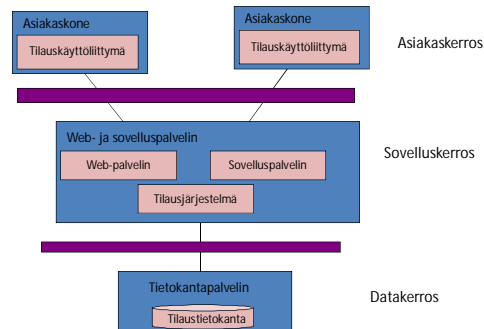
- Erilaiset tai yhteenkuulumattomat *vastuut* (responsibilities) on erotettava toisistaan ohjelmistossa
 - Jaettava eri komponenteille; jokaisella k:lla selkeä tehtävä (vrt. Single Responsibility -periaate)
 - Vastuu: jotakin mitä komponentti *tekee* tai *tietää* tai *piilottaa* muilta (toiminto, riippuvuus, data, ...)
- Tietyn tehtävän yhteistyössä suorittavat komponentit on pidettävä erillään komponenteista, jotka suorittavat muita tehtäviä
- Jos komponentilla on useita *rooleja* eri tilanteita ja käyttöyhteyksiä varten, roolit on pidettävä itsenäisinä ja erillään komponentin sisällä

16.9.2016

581358 Ohjelmistoarkkitehtuurit

17

Miten suunnitteluperiaatteet ilmenevät tässä arkkitehtuurissa?



16.9.2016

581358 Ohjelmistoarkkitehtuurit

18

Coupling

- Kytöntä (coupling) on moduulien välisen assosiaation voimakkuuden mittari
 - Voimakas kytkentä tekee moduulit vaikeammin ymmärrettäväksi, muutettavaksi ja korjattaviksi toisistaan riippumatta
 - Esimerkiksi jos moduulin A luokka A.A pääsee suoraan käsiksi moduulin B luokan B.B toteutukseen (sen kenttiin), syntyy hyvin voimakas kytkentä, koska B.B:n toteutukseen tehtävä muutos todennäköisesti heijastuu välittömästi muutostarpeena A.A:n koodiin, joka käsittelee suoraan B.B:n kenttiä
- Mittarille on tekninen määritelmä, mutta asian *ydin* on miettiä, millaisia *heijastusvaikutuksia* jonkin moduulin sisäisillä muutoksilla on muihin moduuleihin ja pyrkiä minimoimaan ne

https://en.wikipedia.org/wiki/Coupling_%28computer_programming%29

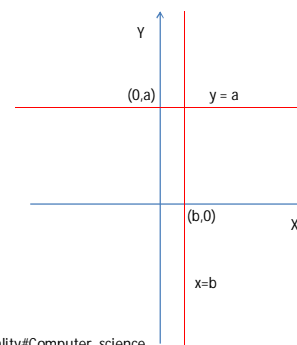
16.9.2016

581358 Ohjelmistoarkkitehtuurit

19

Ortogonalisuus

- Suoralla $y = a$ voi x :n valita vapaasti
 - Jos x :n valinta ei vaikuta y :n arvoon ja päinvastoin, niin ominaisuudet (x ja y) ovat toisistaan riippumattomia
- Suunnittelussa ja arkkitehtuurissa:
 - Moduulien / komponenttien välisen keskinäisen riippumattomuuden mitta



https://en.wikipedia.org/wiki/Orthogonality#Computer_science

16.9.2016

581358 Ohjelmistoarkkitehtuurit

20

Cohesion

- Moduulin sisältämien elementtien yhteenkuuluvuuden (cohesion) mittari
- Yhteenkuuluvuuden eri asteita
 - *Toiminnallinen* (hyvä!): kaikki moduulin elementit toimivat yhdessä jonkin tietyn, rajallisen tehtävän toteuttamiseksi (well-bounded behavior)
 - *Sattumanvarainen* (huono!): moduuli on satunnainen kokoelma yhteenkuulumattomia abstraktioita ja toimintoja
- Asteita on muitakin, mutta oleellista on miettiä, (1) mikä on moduulin päätehtävä ja (2) miten sen elementit liittyvät tuon tehtävän suorittamiseen
 - Voiko jotkut elementit siirtää pois moduulista sen päätehtävän toteuttamisen karsimatta?

https://en.wikipedia.org/wiki/Cohesion_%28computer_science%29

16.9.2016

581358 Ohjelmistoarkkitehtuurit

21

JÄRJESTELMÄN JAKAMINEN OSIIN

16.9.2016

581358 Ohjelmistoarkkitehtuurit

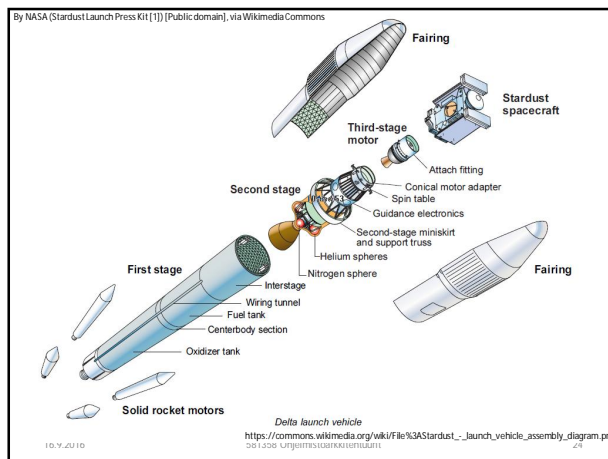
22

How to eat an elephant?

16.9.2016

581358 Ohjelmistoarkkitehtuurit

23



"Legpuzel" by Piero from nl. Licensed under CC BY-SA 3.0 via Wikimedia Commons – <https://commons.wikimedia.org/wiki/File:Legpuzel.JPG#/media/File:Legpuzel.JPG>



16.9.2016

581358 Ohjelmistoarkkitehtuurit

25

Kokonaisuus koostuu osista

- Oletusarkkitehtuuri (esim. ohjelmisto *kehys*, *framework*) antaa valmiin "kaavan", joka jakaa rakennettavan järjestelmän" loogisesti ja fyysisesti erillisiin, tietyn rajatun tehtävän tai tarkoituksen täyttäviin rakennelmiin (moduulit, komponentit jne.)
- Arkkitehtuurityylit ja -patternit tarjoavat suoraan lähtökohia järjestelmän suunnittelulle ja ositukselle
- Yleiset suunnitteluperiaatteet antavat ohjenuoria ja mittareita, jotka auttavat osituksessa ja sen arvioinnissa
- Mitä muita yleisiä tapoja on osittaa järjestelmä ja sen arkkitehtuuri?

(*) SUD = System Under Design

16.9.2016

581358 Ohjelmistoarkkitehtuurit

26

Ositusstrategiat

- *Divide & Conquer*: yhteinen piirre useimmille hyvin suunnitelluille järjestelmille on *hierarkisuus*, jossa ylempien tasojen tarkasteluissa alempien tasojen rakennelmaa (elementtejä) voidaan käsitellä mustina laatikkoina (information hiding) välittämättä niiden sisäisestä rakenteesta
- Sama toistuu rekursiivisesti alemmilla tasoilla

16.9.2016

581358 Ohjelmistoarkkitehtuurit

27

Yleinen ositusstrategia

1. Muodosta abstraktioiden hierarkia, jossa ylempien tasojen elementit (komponentit, moduulit) koostuvat alempien tasojen elementeistä
2. Rajoita elementtien määrä kullakin tasolla korkeintaan muutamaan kymmeneen
3. Jokaisella elementillä on oltava selkeä ja hyvin perusteltu "olemassaolon tarkoitus" (separation of concerns)
4. Noudata tiedon kätkemisen ja kapseloinnin periaatteita (information hiding, encapsulation) niin, että elementit eivät tarpeettomasti paljasta sisäistä toteutustaan

16.9.2016

581358 Ohjelmistoarkkitehtuurit

28

Erityisiä osituksia

- Yleinen strategia ei ota kantaa siihen, *miten* hierarkinen ositus oikein löydetään
- Jotta järjestelmän arkkitehtuuri pysyy ymmärrettävänä ja hallittavana, on yleensä syytä valita yksi *vallitseva ositusstrategia*, jota pyritään johdonmukaisesti noudattamaan
- Kurssikirjassa Fairbanks esittää seuraavat erityiset ositusstrategiat

16.9.2016

581358 Ohjelmistoarkkitehtuurit

29

Ositusstrategiat

- Jako toiminnallisuuden mukaan
 - Tarkastellaan, mitä kaikkea järjestelmän pitää tehdä (toiminnallisuus)
 - Yhteenkuuluva toiminnallisuus yhteen elementtiin tai samalla hierarkian tasolla läheisessä yhteistoiminnassa oleviin elementteihin
 - Toiminnallisuksien jaottelu joko teknisen luonteen tai toiminnan tarkoituksen (sovellusalueen prosessien) perusteella

16.9.2016

581358 Ohjelmistoarkkitehtuurit

30

Ositusstrategiat

- Jako arkkityyppien mukaan
 - Arkkityyppi (archetype) tarkoittaa jotakin sovellusalueen keskeistä käsitettä, esimerkiksi tietomallissa esiintyvää pysyväisluontoista oliota
- Jako arkkitehtuurityylin mukaan
 - Tämä onkin jo tuttua huttua
- Jako tiettyjen laatuvaatimusten saavuttamiseen tähtäävien suunnittelutaktiikoiden perusteella
 - (Quality) Attribute Driven Design
 - Eri laatuattribuutteja varten on omat taktiikkansa

16.9.2016

581358 Ohjelmistoarkkitehtuurit

31

Ositusstrategiat

- Jako järjestelmän sisältämien rajapintojen ja palvelujen mukaan
 - Jokaista palvelua/rajapintaa kohden yksi sen implementoiva komponentti
 - Yksi komponentti per palvelu ei useinkaan riitä, mutta tällä pääsee liikkeelle
- Palapeli
 - Kokonaisuus sovittelaa jo olemassaolevista (toteutetuista) elementeistä erilaisilla sovittimilla ja "liimakomponenteilla"
 - Käytännössä yleinen ratkaisu laajoissa yritysjärjestelmissä, jossa harvoin päästään puhtaalta pöydältä liikkeelle

16.9.2016

581358 Ohjelmistoarkkitehtuurit

32

Ositusstrategiat

- Ongelman uudelleenmuotoilu toisen sovellusalueen käsittein ja valmiin osituksen käyttö
 - Joskus voi olla mahdollista pukea suunnitteluongelma jonkin toisen sovellusalueen vaatimaan muotoon ja käyttää tälle alueelle suunniteltua ositusta
 - Esimerkiksi järjestelmän näkeminen sovellusalueen olioiden muodostaman suunnattuna verkkona ja yleisten verkkoalgoritmien käyttöä toimintojen toteutukseen

16.9.2016

581358 Ohjelmistoarkkitehtuurit

33

KAUNIS ARKKITEHTUURI

16.9.2016

581358 Ohjelmistoarkkitehtuurit

34

"Kaunis arkkitehtuuri"

- Stephen J. Mellor, teoksessa Beautiful Architecture. D. Spinellis, G. Gousios (eds.), O'Reilly, 2009.
- *One fact in one place*
 - Pyritään lokalisoimaan useassa eri yhteydessä tarvittava data tai toiminnallisuus yhteen ainoaan paikkaan
- *Automatic propagation*
 - Lokalisoidun "faktan" kopiointi suoritusaikana käyttökontekstiinsa on joskus tarpeen (saman komponentin instansiointi ja konfigurointi eri palveluissa)
 - Tamän pitäisi tapahtua automaattisesti ja työkalun tukemana (esim *dependency injection*)
- *Architecture includes construction*
 - Ohjelmiston koostaminen ja rakentaminen (build) pitää ottaa huomioon arkkitehtuurissa
 - Esimerkiksi reflektio on voimakas mekanismi, jota kannattaa hyödyntää paitsi suoritusaikana myös suoritettavaa ohjelmaa koostettaessa (esim. *convention over configuration*-periaate)
- *Minimize mechanisms*
 - Periaatteessa saman asian tekeminen erilaisten mekanismien määrä karsittava minimiin (-> 1 kpl)
 - Riittävän hyvä ratkaisu kertaalleen toteuteltuna on parempi kuin kymmenen erillistä joka tilanteeseen "parasta" ratkaisua (vrt. *conceptual integrity*)

16.9.2016

581358 Ohjelmistoarkkitehtuurit

35

"Kaunis arkkitehtuuri"

- *Construct engines*
 - Moottori on *virtuaalikone*, joka tarjoaa geneerisen rajapinnan palveluihinsa
 - Rajapinnan palvelut eivät suoraan vastaa sitä käyttävien komponenttien toteuttamia käyttötappauksia vaan ovat luonteeltaan niitä primitiivisempiä ja yleiskäyttöisempiä
 - Yleistä kerrosarkkitehtuureissa
- *O(G), the order of growth*
 - Ota huomioon järjestelmän todennäköinen kasvu
 - Se mikä toimii pienessä järjestelmässä, ei välttämättä toimi isommassa
- *Resist entropy*
 - Pyri pitämään arkkitehtuuri eheänä ja kirrkaana järjestelmän koko elinkaaren ajan (vältä "rikottuja ikkunoita")
 - Työkaluilla on tärkeä rooli

16.9.2016

581358 Ohjelmistoarkkitehtuurit

36

"Beautiful architectures do more with less"