

Ohjelmistoarkkitehtuurin suunnittelu

Luento 3

10.9.2013

581358 Ohjelmistoarkkitehtuurit

1

Oppimistavoitteet

- Rationaalinen ja empiirinen suunnittelumenetelmä
- Arkkitehtuuritietämyksen lähteet

10.9.2013

581358 Ohjelmistoarkkitehtuurit

2

Miten suunnittelijat suunnittelevat?

RATIONAALINEN JA EMPIIRINEN SUUNNITTELUMENETELMÄ

10.9.2013

581358 Ohjelmistoarkkitehtuurit

3

Suunnittelusta

- Tarkastellaan ohjelmistoarkkitehtuurin suunnittelumenetelmiä eli metodologioita
 - Miten asiantuntija ratkoo suunnitteluongelmia?
 - Millaisia suunnittelutekniikoita tai -strategioita hän käyttää ajattelutyössään?
 - Miten työ etenee ongelman määrittelystä sen ratkaisuun?
- Tämä osuus perustuu pääasiassa seuraaviin lähteisiin
 - Brooks, F. P. JR.: *The Design of Design*. Addison Wesley, Pearson Education, 2010.
 - Buschmann, F.: Learning from Failure, Part 2: Featuritis, Performatitis, and Other Diseases. *Software, IEEE* , vol.27, no.1, pp.10,11, Jan.-Feb. 2010

10.9.2013

581358 Ohjelmistoarkkitehtuurit

4

Suunnittelun rationaalinen malli

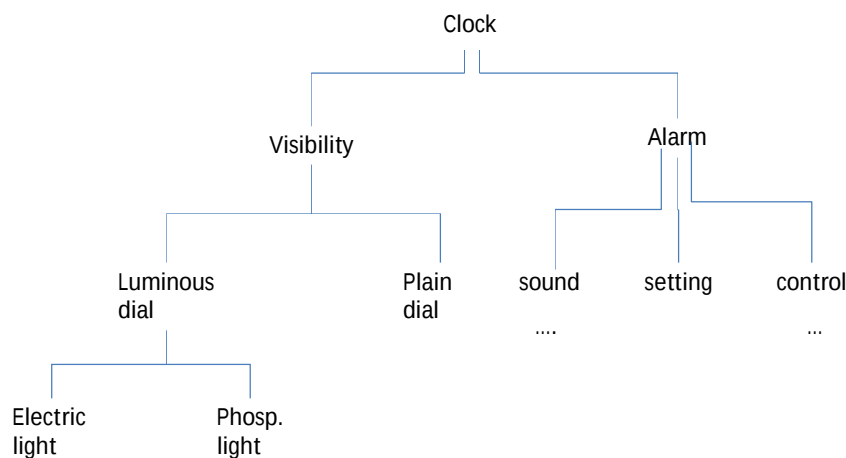
- Lähtökohtana on oletus, että suunnitteluongelma ja ongelman ratkaisun vaatimukset, tavoitteet ja rajoitteet voidaan määritellä tarkasti etukäteen
- Itse suunnittelutyö on etsintää, jossa järjestelmällisesti generoidaan mahdollisia ratkaisuja, evaluoidaan niiden arvo, ja valitaan paras
- Ratkaisu kuvataan puumaisena rakenteena (design tree), jossa kukin solmu vastaa jotain suunnittelupäätöstä ja sen alapuolella oleva puun osa tämän päätöksen jälkeen jäljelle jääviä mahdollisia suunnittelupäätöksiä
 - Kaikki mahdolliset päätöspuut muodostavat yhdessä ratkaisuavaruuden, josta optimaalinen ratkaisu (puu) etsitään

10.9.2013

581358 Ohjelmistoarkkitehtuurit

5

Herätyskellon osittainen päätöspuu



10.9.2013

581358 Ohjelmistoarkkitehtuurit

6

Rationaalinen suunnittelumetodi

- Goal (primääritavoite)
- Desiderata (sekundääriset tavoitteet)
- Utility function (ratkaisun hyödyllisyyden ja hyvyyden mittaava arvofunktio)
- Constraints (budget, resource allocations)
- Design tree decisions
 - UNTIL ("good enough") or (time runs out)
 - DO another design (to improve utility function)
 - UNTIL design is complete // design == tree
 - » WHILE design remains feasible, make another design decision // constraints
 - » END WHILE
 - » Backtrack up design tree
 - » Explore a path not searched before
 - END UNTIL
 - END DO
 - Take best design
 - END UNTIL

10.9.2013

581358 Ohjelmistoarkkitehtuurit

7

Rationaalinen malli

- Rationalisti uskoo, että saatuaan oikean koulutuksen, kasvattavaa kokemusta ja tekemällä riittävän paljon tarpeeksi huolellista ajatustyötä, suunnittelija voi tehdä virheettömän suunnitelman
- Suunnittelumetodi tähtää virheettömyyden saavuttamiseen suunnitteluvaiheessa

10.9.2013

581358 Ohjelmistoarkkitehtuurit

8

Rationaalisen mallin kritiikkiä¹

- Tavoitteet ja vaatimukset ovat harvoin riittävän tarkasti selvillä etukäteen ja ne muuttuvat
 - "We don't really know the goal when we start"
 - Koskee erityisesti tuotesuunnittelua
- Suunnittelupuun rakenne on ongelmallinen
 - Puun rakennetta ei yleensä tunneta etukäteen, vaan se löytyy suunnittelutyön aikana; päätökset avaavat uusia aiemmin tuntemattomia mahdollisuuksia
 - Kombinatorinen räjähdys: suunnittelupäätösten järjestys voi vaikuttaa radikaalisti puun rakenteeseen; yksittäisillä päätöksillä saattaa riippuvuuksien ja sivuvaikutusten kautta olla globaalia vaikutusta

¹Brooks, F. P. JR.: *The Design of Design*. Addison Wesley, Pearson Education, 2010.

Rationaalisen mallin kritiikkiä

- Hyvyysfunktion (utility function) inkrementaalinen evaluointi ei aina ole mahdollista
 - Sen voi evaluoida vasta kun koko päätöspuu on valmis lehtiä myöten
- Suunnittelijat eivät vain tee työtään näin...
 - Käytännössä suunnittelutyö näyttää oskilloivan osaongelma-alueiden ja –ratkaisujen välillä sekä ongelman osittamisen ja osaratkaisujen koostamisen välillä (top-down & bottom-up)
 - Aloittelevalle suunnittelijalle rationaalinen malli voi kuitenkin antaa jonkinlaisen lähtöpisteen työhönsä

Empiirinen malli¹

- Ongelmia ei pystytä määrittelemään täysin etukäteen eikä ratkaisuehdotuksia evaluoimaan ilman *konkretiaa*
 - Ihminen *ei pysty* virheettömän suunnitelman tuottamiseen
 - suunnittelun viat löydetään *kokeellisesti*
- *Suunnittelu ja toteutus* etenevät rinnakkain (iteroiden)
 - Ratkaistaan ensin joitakin ongelmia
 - Testataan ratkaisujen toimivuus
 - Katsotaan, mihin uusiin avauksiin ja mahdollisuuksiin tehdyt ratkaisut johtavat
- *"Design isn't simply selecting from alternatives, but also realizing their existence"*

¹Brooks, F. P. JR.: *The Design of Design*. Addison Wesley, Pearson Education, 2010.

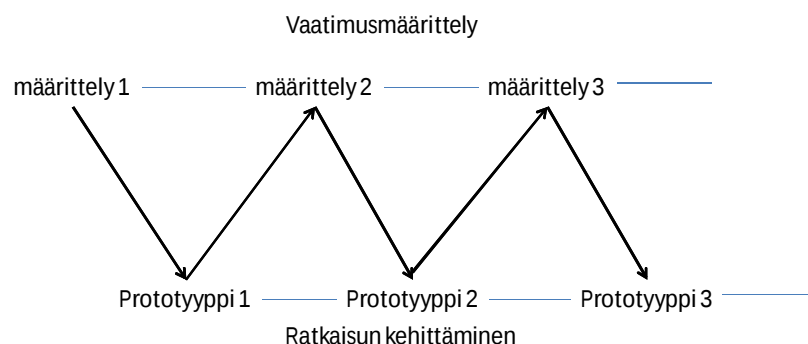
10.9.2013

581358 Ohjelmistoarkkitehtuurit

11

Empiirisiä malleja

- Co-evolutiivinen malli¹



¹ Maher M., Tang H.-H.: Co-evolution as a computational and cognitive model of design. *Research in Engineering Design* Volume 14, Issue 1, 2003, pp 47-64.

10.9.2013

581358 Ohjelmistoarkkitehtuurit

12

Empiirisiä malleja

- Open Source –kehitysmenetelmät (Raymondin ”Bazaar”)
 - Evolutiivinen, ohjelmiston kasvattamiseen pyrkivä malli
 - Kuka tahansa jonkin tarpeen näkevä voi tarjota siihen ratkaisua
 - Avoin kilpailu vaihtoehtoisten ratkaisujen välillä (ála evoluutio)
 - Joukkotestaus, paremman bugikorjaukset
 - Toimii hyvin, kun suunnittelijat ja toteuttajat ovat itse myös käyttäjiä (vaatimukset, ratkaisujen evaluointikriteerit)

10.9.2013

581358 Ohjelmistoarkkitehtuurit

13

Empiirisiä malleja

- Spiral model (Boehm)
 - Käytiin läpi edellisellä luennolla
 - Inkrementaalisuus & kasvattaminen, prototyypit, riskien hallinta

10.9.2013

581358 Ohjelmistoarkkitehtuurit

14

Suunnittelua tiimityönä?

- Brooks korostaa käsitteellisen eheyden (conceptual integrity) merkitystä
 - Sama asia tehdään vain yhdellä ja samalla tavalla eri puolilla järjestelmää
- Hänen mukaansa yhteistoiminnallinen reaaliaikainen (kollaboratiivinen) suunnittelu ei toimi, mutta parityöskentelyssä on taikaa
 - Suunnittelu vaatii itsenäistä miettimistä ja keskittymistä
 - Reaaliaikainen kollaboraatio toimii suunnitelmien katselmoinnissa, mutta ei itse suunnittelussa
- Jossain määrin ongelmallinen asia *emergenssiä* korostaville ketterille menetelmille
 - Ei yleensä erillistä arkkitehdin roolia

10.9.2013

581358 Ohjelmistoarkkitehtuurit

15

Esimerkki - "Walking skeletons"

- Frank Buschmann selostaa empiirisen koulukunnan näkemyksiä noudattavan lähestymistavan ohjelmistoarkkitehtuurin suunnitteluun *Pragmatic Architect* kolumnissaan¹
 - Tavoitteena erityisesti arkkitehtuurin tarkoituksenmukaisuus ja eräiden ei-toivottujen ilmiöiden välttäminen
 - Ei mene suunnitteluprosessin yksityiskohtiin, mutta antaa suuntaviivat

Buschmann, F.: Learning from Failure, Part 2: Featuritis, Performitis, and Other Diseases. *Software, IEEE*, vol.27, no.1, pp.10,11, Jan.-Feb. 2010

10.9.2013

581358 Ohjelmistoarkkitehtuurit

16

Vältettäviä asioita

- *Featuritis* – toiminnallisuuden toteuttamisen korostaminen laatuominaisuuksien kustannuksella
- *Flexibilitis* – arkkitehtuurin kuormittaminen tarpeettoman laajoilla laajennos-, sovitus- ja konfigurointimekanismeilla
- *Performitis* – projektin loppusuoralla suorituskyky havaitaan huonoksi, mikä johtaa laajamittaiseen paikalliseen optimointiin ja häkkäykseen globaalin strategian puuttuessa
 - ongelmien perimmäisenä syynä usein kaksi edellistä "tautia"

10.9.2013

581358 Ohjelmistoarkkitehtuurit

17

Luuranko

- Walking skeleton – ohjelmiston *perusarkkitehtuuri* (baseline), joka tukee
 - Tärkeimpien arvoa tuottavien toiminnallisuuksien toteuttamista (business case)
 - Haastavimpien laatuvaatimusten toteuttamista
 - Ohjelmistotuoteperheen rakentamista (variability)
- Nämä ovat *arkkitehtuurin kannalta merkittävimpiä vaatimuksia* (Architecturally Significant Requirements)

10.9.2013

581358 Ohjelmistoarkkitehtuurit

18

...joka kävelee

- Ketterässä kehityksessä luuranko on suoritettavaa koodia eli se ”kävelee”
- Empirismi
 - Testaus
 - Ominaisuuksien mittaus
 - Vaatimusten validointi

10.9.2013

581358 Ohjelmistoarkkitehtuurit

19

Sovellusalueen merkitys

- Sovellusalueen käsitteet ja ilmiöt ohjaavat merkittäväällä tavalla perusarkkitehtuurin suunnittelua
 - Arkkitehtuurissa suunniteltujen komponenttien vastuiden ja yhteistoiminnan pitää heijastaa sovellusalueen käsitteitä
- Tämä mahdollistaa vaaditun toiminnallisuuden järkevän toteuttamisen (esim. riittävät tietomallit)
 - Sovellusalueen paikalliset muutokset jäävät todennäköisemmin paikallisiksi myös ohjelmistossa

10.9.2013

581358 Ohjelmistoarkkitehtuurit

20

Käyttötapaukset

- Koko ohjelmiston ”läpi menevät” (end-2-end) käyttötapaukset ja skenaariot ohjaavat perusarkkitehtuurin suunnittelua kattamaan sovellusalueen eri osat
 - Suppea mutta edustava joukko tärkeimmät suunnitteluhaasteet esiin tuovia käyttötapauksia
 - Laatuvaatimusten tulisi tulla esille skenaarioissa
 - Sovellustoiminnallisuuden suhde laatuvaatimukset toteuttavaan infrastruktuuriin

10.9.2013

581358 Ohjelmistoarkkitehtuurit

21

Suunnittelutyön kulku

- Suunnittelu lähtee liikkeelle arkkitehtuuristen vaatimusten (ASR) kannalta relevanttien käyttötapausten peruskuluista (main success & failure scenarios)
- Vasta kun peruskulut toteuttava arkkitehtuuri on *vakaa ja se toimii*, aletaan ottaa harkiten mukaan näiden variaatioita ja laajennoksia seuraavissa iteraatioissa
 - Piirremalleja voidaan käyttää apuna (myöhemmin kurssilla)
- Luurangon ympärille kasvatetaan lihaksia, ei läskiä

10.9.2013

581358 Ohjelmistoarkkitehtuurit

22

Yhteenveto

- Rationaalinen malli ei käytännössä toimi
 - Voi soveltua kuitenkin rajattuihin, matemaattisessa mielessä hyvin määriteltyihin ongelmiin
- Empiiriset menetelmät (iteroivat ja evolutiiviset) sopivat paremmin reaalimaailman projekteihin, joissa erehtyväiset ihmiset toimivat epätäydellisen tiedon varassa
 - Vuorottelu ongelma- ja ratkaisuavaruuden välillä, prototyypit ja validointi
 - Ratkaisujen rakentaminen mahdollista sekä *top-down* (osittamalla, divide & conquer) että *bottom-up* (osaratkaisusta kokoamalla)

10.9.2013

581358 Ohjelmistoarkkitehtuurit

23

ARKKITEHTUURITIEDON LÄHTEET

10.9.2013

581358 Ohjelmistoarkkitehtuurit

24

Ohjelmistoarkkitehtuuritiedon lähteillä

- Yhdellä (yliopisto-) kurssilla ei kenestäkään kouluteta ohjelmistoarkkitehtia
- Ohjelmistoarkkitehdiksi *kasvetaan* kokemuksen ja haastavien tehtävien kautta
- Muitten kokemuksesta voi kuitenkin ottaa oppia ja kehittää suunnittelutaitojaan
 - Arkkitehtuurityylit ja ratkaisumallit
 - Yleiset ohjelmistojen suunnitteluperiaatteet
 - Laatuominaisuuksien suunnittelutaktiikat
 - Handbook of Software Architecture (käsikirjat)

10.9.2013

581358 Ohjelmistoarkkitehtuurit

25

Arkkitehtuurityyli

- *Architectural style*
 - Nimetty kokoelma tiettyyn käyttöyhteyteen soveltuvia yleisiä suunnitteluperiaatteita ja sääntöjä, jotka tuovat mukanaan hyödyllisiä ominaisuuksia rakennettavaan järjestelmään
- Esim. asiakas – palvelin tyyli:
 - 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 niiden määrän dynaaminen lisääminen

10.9.2013

581358 Ohjelmistoarkkitehtuurit

26

Arkkitehtoninen ratkaisumalli

- *Architectural pattern*
 - Nimetty kokoelma johonkin toistuvaan suunnitteluongelmaan soveltuvia suunnitteluratkaisuja, jotka on parametrisoitu ottamaan huomioon käyttöyhteys jossa ongelma esiintyy
- Miten tyyli eroaa ratkaisumallista?
 - Tyylin käyttötilanne on yleisempi, ratkaisumallin konkreettisempi
 - Tyylit ovat enemmän periaatesääntöjä ratkaisumallit konkreettisia ratkaisuja
 - Ratkaisumallit soveltavat tyyliä (tyylejä)
 - Ratkaisumalliin voidaan yhdistää tyylejä. Tietty tyyliä noudattava ratkaisu voi sisältää useita ratkaisumalleja.
 - Kaikki lähteet eivät erottele näitä käsitteitä

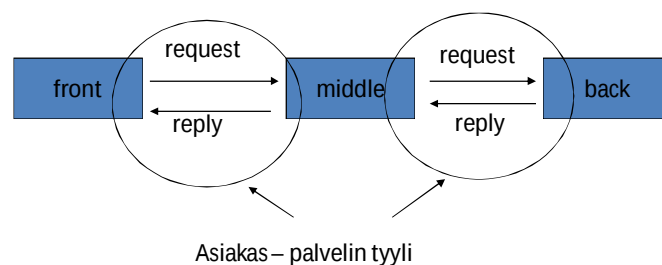
10.9.2013

581358 Ohjelmistoarkkitehtuurit

27

Tyyli ja malli

- Esimerkki:
 - Kolmitasomalli – arkkitehtoninen ratkaisumalli

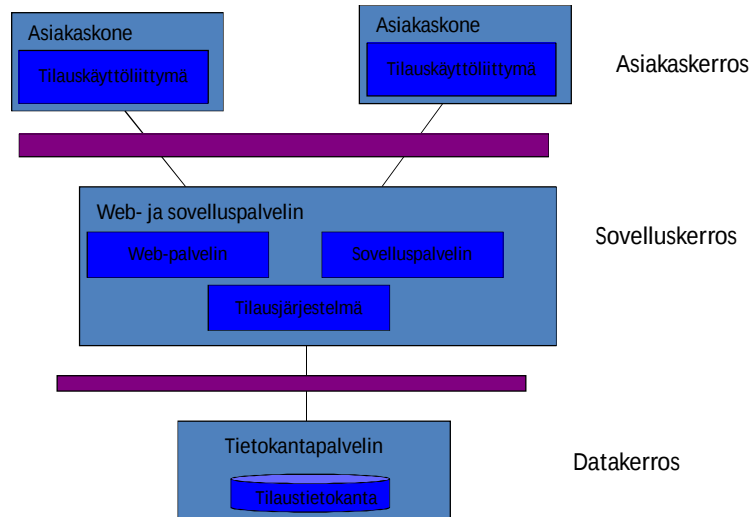


10.9.2013

581358 Ohjelmistoarkkitehtuurit

28

Kolmitasomalli



10.9.2013

581358 Ohjelmistoarkkitehtuurit

29

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 seuraavalla luennolla

10.9.2013

581358 Ohjelmistoarkkitehtuurit

30

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

10.9.2013

581358 Ohjelmistoarkkitehtuurit

31

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 implementaation tai sovellusalueen piirteen toteutuksen (= suunnittelupäätökset)
 - Moduuli tarjoaa palveluihinsa vakaan abstraktin rajapinnan, joka ei paljasta toteutuksen yksityiskohtia (~abstrakti tietotyyppi)

10.9.2013

581358 Ohjelmistoarkkitehtuurit

32

Information Hiding

- Kun moduulin kätkevät suunnittelupäätökset muuttuvat, moduulien käyttäjiä ei tarvitse muuttaa, koska ne riippuvat vain samana pysyvästä rajapinnasta
- Useimmat ohjelmistokehittäjät pitävät itsestäänselvyytenä, että rajapinta ei paljasta esim. olion toteutuksen yksityiskohtia
- Harvempi kuitenkaan tulee ajatelleeksi *odotettavissa olevia muutoksia ja niiden heijastusvaikutusten proaktiivista estämistä* ohjelmiston modularisoinnin kautta
 - Arkkitehdin työn kuvaan tällaisen ajattelemisen kuitenkin kuuluu