

Arkkitehtuurin mallintaminen

Luento 8

27.9.2016

581385 Ohjelmistoarkkitehtuurit

1

Oppimistavoitteet

- Miksi ja milloin ohjelmistoarkkitehtuurista kannattaa laatia malleja?
- Ohjelmistoarkkitehtuurin mallin ohjeellinen rakenne

27.9.2016

581385 Ohjelmistoarkkitehtuurit

2

MIKSI MALLEJA?

27.9.2016

581385 Ohjelmistoarkkitehtuurit

3

Malleista ja niiden käytöstä

- ¹Malli =
esine, kuva, kuvio, kaava tms., jonka mukaan tehdään jtk t. josta havainnoidaan jtk. [...]
- Erik.*
[...]
- b. jtk kuviona, kaaviona tm. havainnollistava esitys.
Molekyylin kolmiulotteinen malli. Atomiytimen malli. Väestön kehitystä kuvaava matemaattinen malli.
- [...]

¹MOT Sanakirja

27.9.2016

581385 Ohjelmistoarkkitehtuurit

4

Malleista ja niiden käytöstä

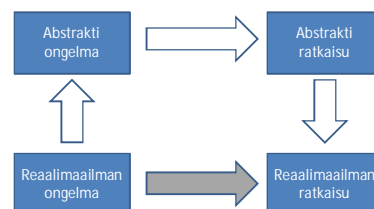
- Yksinkertaiset ongelmat voidaan ratkaista suoraan (ratkaisu on ilmeinen)
- Monimutkaiset reaali maailman ongelmat vaativat asian kuvaamista abstraktina mallina (joka pelkistää ongelman ytimen) ja sen ratkaisemista ensin mallissa, minkä jälkeen ratkaisu siirretään reaali maailman implementaatioksi
 - Esimerkiksi matemaattisten mallien käyttö rakenteiden lujuuslaskennassa jne.
 - 3D-mallien käyttö rakennusten suunnittelun apuna

27.9.2016

581385 Ohjelmistoarkkitehtuurit

5

Kommutatiivinen kaavio



27.9.2016

581385 Ohjelmistoarkkitehtuurit

6

Mallien käyttö

- Ongelman laajuus ja monimutkaisuus vaatii abstrahointia (eli yksityiskohtien karsimista)
 - On nähtävä metsä puilta
 - Yksittäistä luokkaa, moduulia tai suunnittelumallin ilmentymää laajempien kokonaisuuksien hahmottaminen ja niiden roolin ymmärtäminen
 - Suuren ohjelmiston koodin lukeminen rivi riviltä sen arkkitehtuurin ymmärtämiseksi on käytännössä mahdotonta

27.9.2016

581385 Ohjelmistoarkkitehtuurit

7

Mallien käyttö

- Abstraktiot *pelkistävät ongelmia* ja tarjoavat mahdollisuuden kehittää *työkaluja*
 - Oikein laadittu abstraktio vangitsee reaali maailman ongelman *oleelliset piirteet*, jotka vaikuttavat hyväksyttävän ratkaisun muodostamiseen
 - 'Turhat' yksityiskohdat karsitaan pois
 - Abstraktiin malliin voidaan käyttää erityisiä työkaluja ongelmien ratkaisemiseksi
 - Esim moduulien/luokkien uudelleensijoittelu kirjastoihin/pakkausihin haitallisiksi havaittujen riippuvuuksien karsimiseksi ja heijastusvaikutusten vähentämiseksi

27.9.2016

581385 Ohjelmistoarkkitehtuurit

8

Mallien käyttö

- Mallit helpottavat päätelmien tekemistä järjestelmän ominaisuuksista
 - Malli auttaa päättämään, mitkä yksityiskohdat ja suunnittelupäätökset ovat tärkeitä *laatuominaisuuksien* ja niiden toteutumisen arvioinnin kannalta
 - Malli voi olla luonnosmainen ja/tai pelkästään analyysoijan mielessä

27.9.2016

581385 Ohjelmistoarkkitehtuurit

9

Web-palvelimen malliluonnos

Asiakkaan yksi web-sivun pyyntö
voi aiheuttaa monia tietokantahakua



Asiakkaan kokema viive pyyntöön vastaamisessa =
Viestinvälitykseen kuluva aika + palvelimen pyynnön prosessointiaika +
(tietokantahakuun kuluva aika x hakujen lukumäärä)

27.9.2016

581385 Ohjelmistoarkkitehtuurit

10

Web-palvelimen malliluonnos

- Kun edellisen mallin suoritusajoille annetaan arviot (esim. $10 + 10 + n \times 25\text{ms}$), niin mallin perusteella on helppo nähdä, että tietokantakyselyihin kuluva aika dominoi asiakkaan kokemaa viivettä
 - Normalisoitu, hierarkinen relaatiotietokanta vs. normalisoimaton "flat" -tietomalli
 - -> Tietokantahakujen määrässä voi olla *kertaluokan ero*
- Malli jättää monia yksityiskohtia huomiotta (cache:t, jonotus*), mutta tällaisenaankin se on hyödyllinen analyysin kannalta

*) samanaikaisten pyyntöjen aiheuttaman jonotuksen ja resurssikilpailun mallintaminen ei ole sekään erityisen vaikeaa oikeilla työkaluilla ja matem. mallinnusmenetelmillä

27.9.2016

581385 Ohjelmistoarkkitehtuurit

11

Mallien käyttö

- Mallit karsivat ongelman ratkaisussa käsiteltäviä yksityiskohtia
 - Malli valikoi ongelman ratkaisemisen kannalta oleelliset asiat ja jättää muut pois
 - Malli on aina jossain mielessä epätäydellinen, mutta reaali maailman "täydellinen malli" olisi liian suuri ja hankala käsiteltäväksi ja käsitettäväksi
 - "Essentially, all models are wrong but some are *useful*."

27.9.2016

581385 Ohjelmistoarkkitehtuurit

12

Mallien käyttö

- Mallit tehostavat päätelmien ja arviointien tekemistä
 - Suunnittelija voi keskittyä vain pieneen joukkoon suunnittelupäätöksiä/yksityiskohtia kerrallaan (=työmuisti ja kognitio)
 - Käsitteellinen tai konkreettinen malli auttaa suhteuttamaan ratkaisun eri yksityiskohtia toisiinsa ja havaitsemaan ja analysoimaan niiden välisiä riippuvuuksia ja rajoitteita

27.9.2016

581385 Ohjelmistoarkkitehtuurit

13

Mallien käyttö

- *Kysy ensin* mitä haluat tietää ja laadi *malli vasta sitten*
 - Ei malleja mallinnuksen vuoksi
 - Samasta asiasta voidaan laatia erilaisia malleja eri käyttötarkoituksiin (suorituskyky, turvallisuus, riippuvuuksien hallinta jne.) !
- Arkkitehtuurityössä malleista on hyötyä ja ne ovat usein jopa välttämättömiä, mutta konkreettisia malleja pitäisi laatia vain *todelliseen tarpeeseen*, ei varmuuden vuoksi

27.9.2016

581385 Ohjelmistoarkkitehtuurit

14

OHJELMISTOARKKITEHTUURIN MALLIT JA NIIDEN RAKENNE

27.9.2016

581385 Ohjelmistoarkkitehtuurit

15

Ohjelmistoarkkitehtuurin mallintaminen

- Fairbanks esittää *kanonisen¹ rakenteen* ohjelmistoarkkitehtuurin malleille
 - Määrittelee minkälaisia asioita ohjelmistoarkkitehtuurin mallin pitäisi yleisesti ottaen kuvata
 - *metamalli* tai *mallinstandardi*
 - Kattaa koko arkkitehtuuriratkaisun aina sovellusalueen käsitteistä toteutuksen koodiin ja laitteistosijoitteluun asti

1) *Ohjeellinen, esikuvallinen* - MOT Kielitoimiston sanakirja

27.9.2016

581385 Ohjelmistoarkkitehtuurit

16

Ohjelmistoarkkitehtuurin mallintaminen

- Kanoninen rakenne antaa myös *käsitteellisen kehysten* (conceptual model) ohjelmistoarkkitehtuurille
 - Kehys tukee OA:n suunnittelua ohjaamalla suunnittelutyötä tietynlaisten kysymysten ja rakenteiden pohtimiseen
 - Ohjaa jakamaan ohjelmistoa koskevien faktojen käsittelyä sopivalle abstraktiotasolle
- Ei ole kuitenkaan tarkoitus, että jokaisesta ohjelmistoprojektista laaditaan kanonisen rakenteen mukainen kattava kuvaus
 - Projektit valikoivat todellisen tarpeen mukaan, mitä osia arkkitehtuurista mallinnetaan kanonisen metamallin sääntöjä noudattaen

27.9.2016

581385 Ohjelmistoarkkitehtuurit

17

Ohjelmistoarkkitehtuurin mallintaminen

- Harhakäsityksiä
 - Jokaisen projektin pitää dokumentoida arkkitehtuurinsa: väärin. Arkkitehtuurimalleja ja dokumentteja kannattaa laatia vain kun ne auttavat pienentämään (projekti- ja tuote-) riskejä
 - Arkkitehtuurin dokumentaation pitää olla aina kattava: väärin. Mallinnuksen pitäisi kohdistua riskipitoisimpiin ratkaisun osa-alueisiin (laatuominaisuuksiin).
 - Laaja dokumentaatiokin saattaa joskus olla tarpeen, esimerkiksi erityisen kommunikointitarpeen tyydyttämiseksi !

27.9.2016

581385 Ohjelmistoarkkitehtuurit

18

Ohjelmistoarkkitehtuurin mallintaminen

- Suunnittelun pitäisi aina edellyttää koodausta: väärin. Projektin aikaisessa vaiheessa tehty ratkaisun osittainen toteutus ja prototypointi voivat auttaa tunnistamaan hankalimmat ongelmat.
 - Vrt. Walking Skeleton

27.9.2016

581385 Ohjelmistoarkkitehtuurit

19

Kanonisen rakenteen perusideat

- Arkkitehtuurin malli (jatkossa a-malli) jakautuu kolmeen osamalliin
 1. Sovellusaluemalli (domain model)
 2. Suunnittelumalli (design model)
 3. Koodimalli (code model)
- Sovellusaluemalli on abstraktein ja koodimalli konkreettisin (lähimpänä toteutusta)
 - *Vastaavuus-* ja *tarkennussuhteet* (designation and refinement) kytkevät osamallit toisiinsa

27.9.2016

581385 Ohjelmistoarkkitehtuurit

20

Kanoninen rakenteen perusideat

- Osamallit voivat periaatteessa olla hyvin laajoja, mutta *näkymillä* (view) voidaan suodattaa ja poimia esiin vain tietyn näkökulman kannalta mielenkiintoisia faktoja
 - Käytännössä a-mallin fyysinen ilmentymä (kaaviokokoelma, dokumentti) koostuu aina tietyin perustein valituista näkymistä
 - ”Täydellinen” malli voi olla olemassa vain suunnittelijoiden mielissä

27.9.2016

581385 Ohjelmistoarkkitehtuurit

21

Kanoninen rakenteen perusideat

- Malli jakaa erilaiset faktat toteutettavasta ohjelmistosta omiin osamalleihinsa
 - ”Laskutusjakso on 30 vuorokautta”
 - ”Ladatut fonttiresurssit pitää aina eksplisiittisesti vapauttaa”
 - ”Asiakkaan osoite on talletettu varchar(80) – tyyppiseen kenttään”
- Auttaa pitämään erityyppiset asiat erillään helpottaen niiden ymmärtämistä ja käsittelemistä

27.9.2016

581385 Ohjelmistoarkkitehtuurit

22

Sovellusaluemalli

- Ilmaisee reaali maailman eli ongelma-alueen tosiasioita, jotka ovat relevantteja toteutettavan ohjelmiston kannalta (~ käsitteet jotka esiintyvät toiminnallisissa vaatimuksissa)
 - Ilmiöt ja oliot
 - Niiden keskinäiset suhteet
 - Miten yllämainitut kehittyvät ja muuttuvat ajan kuluessa
- Näihin tosiasioihin on suhtauduttava *annettuina*; niitä ei voi muuttaa tai tulkita toisin (esim. viikossa on aina seitsemän päivää)

27.9.2016

581385 Ohjelmistoarkkitehtuurit

23

Suunnittelumalli

- Ohjelmistototeutuksen (SUD; System Under Design) *suunnittelu* taas on täysin ohjelmistoprojektin käsissä
- Suunnittelumalli on *osajoukko* kaikista ohjelmiston suunnittelupäätöksistä
 - Osa päätöksistä jää avoimiksi ja koodimallissa ratkaistaviksi
- Se koostuu rekursiivisesti sisäkkäisistä rajapinta- (boundary) ja sisärakennemalleista
 - Hierarkia, yksityiskohtaisuuden tasot !
 - Rajapintamalli kuvaa elementin julkisen rajapinnan muuhun ohjelmistoon
 - Sisärakennemalli kuvaa elementin yksityisen sisäisen suunnittelun

27.9.2016

581385 Ohjelmistoarkkitehtuurit

24

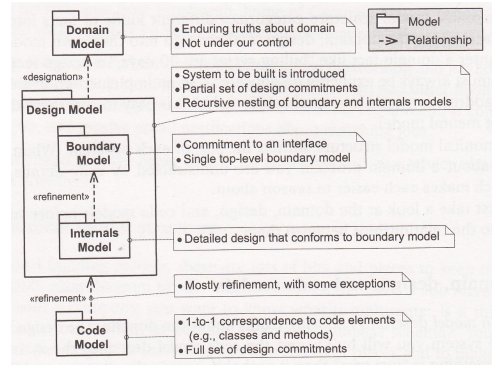
Koodimalli

- Koodimalli on toteutetun ohjelmiston lähdekoodi tai jokin muu *toteutuksen kanssa ekvivalentti* malli
 - Voidaan tuottaa käänteistekniikalla (reverse-engineering, code-to-UML)
 - Siinä missä suunnittelumalli jättää joitain suunnittelupäätöksiä avoimiksi, koodimalli on riittävän *täydellinen suoritettavaksi tietokoneessa*

27.9.2016

581385 Ohjelmistoarkkitehtuurit

25



Fairbanks G.: Just Enough Software Architecture, pp. 116

27.9.2016

581385 Ohjelmistoarkkitehtuurit

26

Vastaavuussuhde

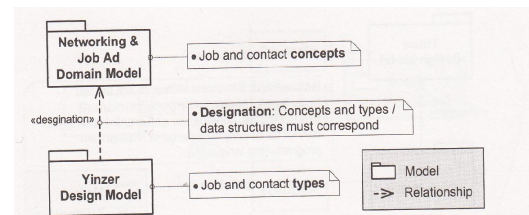
- Vastaavuussuhde linkittää kahden eri mallin samanlaiset oliot toisiinsa
- Sovellusaluemallin käsitteillä ja ilmiöillä on oltava vastinparinsa suunnittelumallissa
- Vastaavuussuhdetta ei välttämättä määritellä eksplisiittisesti (listaamalla vastaavuuksia, mapping), mutta se on voitava aina validoida
 - Vastaavuuden rikkoutuminen johtaa yleensä ohjelmistovikoihin, jotka näyttäytyvät vääränä käyttäytymisenä (hyväksyntä-) testauksessa
 - Vastaavuus ei välttämättä ole 100% oikein toimivassakaan ohjelmistossa, koska suunnittelussa voidaan tarkoituksella tehdä yksinkertaisuuksia sovellusalueen tietotyyppeihin

27.9.2016

581385 Ohjelmistoarkkitehtuurit

27

Vastaavuussuhde



Fairbanks G.: Just Enough Software Architecture, pp. 117

27.9.2016

581385 Ohjelmistoarkkitehtuurit

28

Tarkennussuhde

- Tarkennus kytkee toisiinsa saman olion vähän yksityiskohtia sisältävän (low-detail) mallin ja yksityiskohtaisen (high-detail) mallin
- Sitä käytetään kytkemään rajapintamalli sisärakennemalliin, joka siis kuvaa elementin julkisen rajapinnan toteutuksen (suunnittelun tasolla)
- Tarkennusta voidaan käyttää myös hierarkisesti jakamaan jokin kokonaisuus osiinsa

27.9.2016

581385 Ohjelmistoarkkitehtuurit

29

Tarkennussuhde

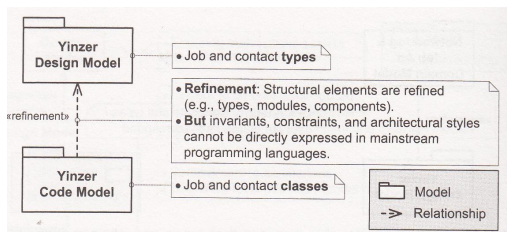
- Tarkennusta käytetään myös linkittämään suunnittelumalli koodimalliin
- Suunnittelumallin rakenteelliset elementit kuvautuvat yleensä suoraan koodimallin elementteihin
 - Moduuli suunnittelumallissa vastaa pakkausta koodissa
 - Komponentti suunnittelussa vastaa joukkoa luokkia (niiden ilmentymiä eli olioita) koodissa
- Mutta on kuitenkin joukko suunnittelumallin elementtejä, joilla ei ole suoraa vastinetta koodissa
 - Invariantit, rajoitteet, arkkitehtuurityylit ja ratkaisumallit
 - Koodissa voi pyrkiä noudattamaan niitä, mutta niitä ei voi suoraan ilmaista ohjelmointikielillä (kieli ei määrittele)

27.9.2016

581385 Ohjelmistoarkkitehtuurit

30

Tarkennussuhde



Fairbanks G.: Just Enough Software Architecture, pp. 118

27.9.2016

581385 Ohjelmistoarkkitehtuurit

31

Näkymät

- Sovellusalue-, suunnittelu- ja koodimallit ovat täynnä yksityiskohtia (ainakin käsitteellisessä mielessä), joita on hankalaa pitää mielessä yhtä aikaa saati dokumentoida eksplisiittisesti
- *Näkymä* eli *projektio* suodattaa mallista osajoukon yksityiskohtia jotakin tiettyä näkökulmaa (viewpoint) tai tarvetta varten

27.9.2016

581385 Ohjelmistoarkkitehtuurit

32

Näkymät

- Esimerkiksi kaupungin maa-alue ja sen rakennukset, tiet ym. muodostavat mallin
 - *Opaskartta* auttaa asukasta löytämään tietyn katuosoitteen kaupungista
 - *Reittiopas* kertoo julkisten liikennevälineiden reitit kaupungissa
 - *Rakentajat ja kaavoittajat* tarvitsevat taas aivan erilaisia tietoja omiin karttoihinsa

27.9.2016

581385 Ohjelmistoarkkitehtuurit

33

Näkymät

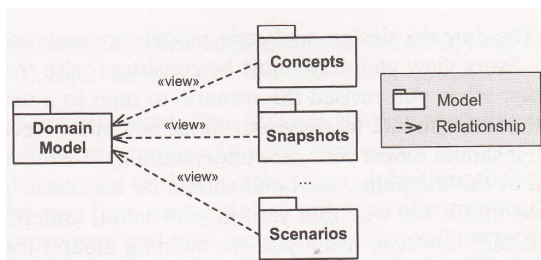
- Eri näkymät samaan "Master" -malliin eivät saisi olla keskenään ristiriitaisia
 - Jos jokin sovellusaluemallin toiminnallinen skenaario viittaa johonkin oliotyyppiin, tyyppi pitäisi olla kuvattuna myös mallin tietomallissa
- "Master" -mallia ei välttämättä koskaan täydellisenä dokumentoida, joten on tärkeää muuten huolehtia, että kaikilla projektiin osallistuvilla on sama "master" -malli mielessään

27.9.2016

581385 Ohjelmistoarkkitehtuurit

34

"Master model"



Fairbanks G.: Just Enough Software Architecture, pp. 119

27.9.2016

581385 Ohjelmistoarkkitehtuurit

35

UML

- UML 2.0 -versioon on lisätty tukea arkkitehtuurien mallinnukselle
- Kieli on laaja ja sen semantiikka ei ole kovin hyvin määritelty, mutta sille on melko yleinen hyväksyntä ja hyvä, joskin kirjava, työkalutuki
- Kurssilla ja kurssikirjassa käytetään UML 2.0:aa mallien visualisointiin

27.9.2016

581385 Ohjelmistoarkkitehtuurit

36

UML -työkalut

- Täysveriset UML-työkalut ovat isoja ja raskaskäyttöisiä (ja usein myös kalliita) ohjelmistoja
 - Tukevat täydellisten mallien laatimista ja esim. automaattista koodin generointia
- Kevyempiä välineitä tämän kurssin tarpeisiin
 - Draw.io <https://www.draw.io/>, varsin monipuolinen ilmainen piirrosväline, joka ei aivan täysin tue UML 2.0:n rakennekaavioiden (composite structure diagram) piirtämistä, mutta pienellä saatamisella nekin onnistuvat
 - UMLet <http://www.umlet.com/> - osin tekstipohjainen, osin graafinen, askeettinen mutta suhteellisen toimiva väline, jolla onnistuvat myös UML 2.0:n arkkitehtuuritason rakennekaaviot (composite structure diagram)
 - PlantUML, <http://plantuml.sourceforge.net/> - helppokäyttöinen tekstipohjainen UML-työkalu, ei tukea UML 2.0:n arkkitehtuuritason rakennekaavioille (composite structure diagram)
- Verkossa toimivia piirtovälineitä (ominaisuuksiltaan rajoituneempia)
 - <http://uml.me/> (tuokka- ja käyttötapauskaavioihin)
 - <https://www.websequencediagrams.com/> (sekvenssikaavioihin)
- Laitoksen Linux-koneilta löytyy *Umbrello*, joka ei kuitenkaan tue UML 2.0:n rakennekaavioita (komponenttikaavioita voi piirtää)