

## Suunnittelumalli

### *Design Model*

Luento 9

29.9.2015

581385 Ohjelmistoarkkitehtuurit

1

## Oppimistavoitteet

- Suunnittelumalli
  - Rajapintamalli, sisärakennemallit

29.9.2015

581385 Ohjelmistoarkkitehtuurit

2

## SUUNNITTELUMALLI

29.9.2015

581385 Ohjelmistoarkkitehtuurit

3

## Suunnittelumalli

- Suunnittelumalli sisältää arkkitehtuurin suunnittelun ja analysoinnin kannalta oleelliset *suunnittelupäätökset*
  - Käsitteellisellä tasolla se on täydellinen sisältäen *kaikki* oleelliset suunnittelupäätökset (Master-malli)
- Jos laadittaisiin *konkreettinen* täydellinen suunnittelumalli, siitä tulisi hyvin laaja ja siksi epäkäytännöllinen
  - Ymmärrettävyys, ylläpidettävyys

29.9.2015

581385 Ohjelmistoarkkitehtuurit

4

## Suunnittelumalli

- Täydellinen suunnittelumalli jääköön siis suunnittelijoiden mieleen, mutta siitä on silti pystyttävä laatimaan *konkreettisia kuvauksia*, joita voidaan *efektiivisesti käyttää* päätelmien tekoon arkkitehtuurista
- Seuraavia tekniikoita käytetään efektiivisten suunnittelumallien laadinnassa
  - Näytetään vain ne yksityiskohdat, joita tarvitaan suunnitteluongelmien ratkomiseen

29.9.2015

581385 Ohjelmistoarkkitehtuurit

5

## Suunnittelumalli

- Näkymä (*View*)
  - Projektiio, joka poimii tietyn näkökulman mukaisen joukon yksityiskohtia täydellisestä Master-mallista konkreettiseksi kuvaukseksi
- Kapselointi (*Encapsulation*)
  - Arkkitehtuurielementin malli jaetaan *rajapintamalliin* (boundary model) ja *sisärakennemalliin* (internals model)
  - Erotetaan elementin julkisesti muille elementeille näkyvä osa (rajapinta, palvelut) sen toteutuksesta

29.9.2015

581385 Ohjelmistoarkkitehtuurit

6

## Suunnittelumalli

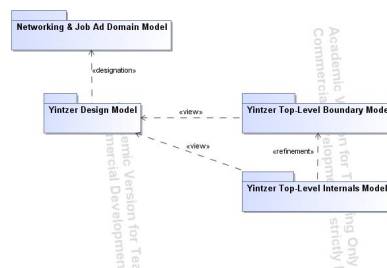
- Sisäkkäisyys (*Nesting*)
  - Useimmilla mallin elementeillä on hienorakenne (sub-structure)
  - Sisärakennemalli kuvaa hienorakenteen, joka esitetään sisältyvien komponenttien rajapintamallin avulla
  - Sisältyvillä elementeillä on taas oma sisäinen rakenteensa
  - Näin syntyy puumainen elementtihierarkia

29.9.2015

581385 Ohjelmistoarkkitehtuurit

7

## Yinzer Top-Level Design Model



29.9.2015

581385 Ohjelmistoarkkitehtuurit

8

## Yinzer-palvelun suunnittelumalli

- Seuraavassa käydään läpi *Yinzer –SoMe-* palvelun suunnittelumalli
  - Ylimmän tason rajapintamalli, joka kuvaa koko järjestelmän rajapinnan muuhun maailmaan
    - Käyttötapaukset ja toiminnalliset skenaariot, systeemikonteksti, ulos näkyvät koodimoduulit, sijoittelu laitteistoon, laatuskenaariot
  - Ylimmän tason sisärakennemalli, joka kuvaa koko järjestelmän sisäistä suunnittelua
    - Komponenttikokoonpano, sisäkkäiset rakennemallit

29.9.2015

581385 Ohjelmistoarkkitehtuurit

9

## YLIMMÄN TASON RAJAPINTAMALLI

29.9.2015

581385 Ohjelmistoarkkitehtuurit

10

Ylimmän tason rajapintamalli

## KÄYTTÖTAPAUKSET

29.9.2015

581385 Ohjelmistoarkkitehtuurit

11

## Rajapintamalli

### Käyttötapaukset ja skenaariot

- UML:n *käyttötapauskaaviot* (use case diagram) tarjoavat kompaktin graafisen yhteenvedon
  - Järjestelmän toiminnallisuudesta
  - Toimijoista (Actor) ja muista järjestelmistä, joiden kanssa SUD<sup>1</sup> on vuorovaikutuksessa
- Kaavion käyttötapaukset kuvaavat järjestelmän toiminnallisuuden yleisellä tasolla, eivät mitään tiettyä yhtä skenaariota eivätkä käyttötapausten keskinäisiä riippuvuuksia (esim. suoritusjärjestys)

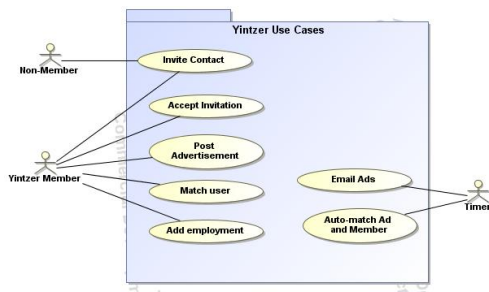
<sup>1</sup>SUD = system under design, suunnitteilla oleva järjestelmä

29.9.2015

581385 Ohjelmistoarkkitehtuurit

12

## Käyttötapauskaavio



29.9.2015

581385 Ohjelmistoarkkitehtuurit

13

Yliimän tason rajapintamalli

## TOIMINNALLISET SKENAARIOT

29.9.2015

581385 Ohjelmistoarkkitehtuurit

14

## Toiminnalliset skenaariot

- *Skenaariot* kuvaavat täydellisiä tapahtumakulkuja, joissa SUD on vuorovaikutuksessa ulkopuolisten toimijoiden (actor) kanssa
  - Erona sovellusalueen skenaarioihin nyt voidaan jo puhua *Yintzer-järjestelmästä* ja siihen kuuluvista jäsenistä (member) erotuksena muista ihmisistä
  - Skenaarioiden kuluksa voidaan jo ottaa kantaa myös suunnitteluratkaisuihin, eli *miten* järjestelmän toiminta näkyy *järjestelmän rajapinnalla* ja *vuorovaikutuksessa* ulkop. toimijoiden kanssa
    - *ei* kuvata siis SUD:n *sisäistä* toiminnan kulkua

29.9.2015

581385 Ohjelmistoarkkitehtuurit

15

## Skenaarioesimerkki

| Nimi     | Kevin saa toita Widgetron-yhtiöstä   |
|----------|--|
| Alkutila | Alan ja Owen ovat jäseniä Yintzerissä; Kevin ei ole jäsen. Alan on toisissa Widgetronissa  |
| Toimijat | Alan, Kevin, Owen, Widgetron   |
| Askelet  | <ol style="list-style-type: none"> <li>1. Alan kutsuu Kevinin kontaktikseen (Contact) omaan verkostoonsa (Network). / Jarjestelmä lähettää Kevinille kutsun sähköpostissa.</li> <li>2. Kevin klikkaa sähköpostiviestissä olevaa linkkiä, liittyy jäseneksi Yintzeriin ja hyväksyy Alanin kutsuun liittyvän kontaktipyynnön.</li> <li>3. Widgetron julkaisee ilmoituksen (Ad) avoimesta ohjelmistokehittäjän toimesta (Job). / Jarjestelmä automaattisesti ehdottaa Owenia rekrytoitavaksi ja lähettää hänelle sähköpostin.</li> <li>4. Alan näkee ilmoituksen ja ehdottaa Kevinia toimeen. / Jarjestelmä lähettää Kevinille sähköpostin.</li> <li>5. Kevin saa työpaikan ja hän päivittää Yintzer-profiiliinsa uuden työnsä Widgetron-yhtiössä.</li> </ol> |

29.9.2015

581385 Ohjelmistoarkkitehtuurit

16

## Skenaarioesimerkki

- Jokainen edellä olevan skenaarioiden erillinen askel vastaa yhtä käyttötapausmallin käyttötapausta
  - Käyttötapausmalli kuvaa mahdolliset tapaukset, mutta skenaario kuvaa jonkin tietyn ketjun käyttötapausten suorituksia
  - Skenaarioiden ei tarvitse käyttää kaikkia käyttötapausta; tämä oli vain yksi esimerkki
- Jos käyttötapausten keskinäiset riippuvuudet ovat tärkeitä (esim suoritusjärjestys), UML:n aktiviteettikaaviot sopivat kaikkien mahdollisten laillisten polkujen (skenaarioiden) määrittelyyn yleisen (tila-)mallin avulla

29.9.2015

581385 Ohjelmistoarkkitehtuurit

17

## Skenaarioesimerkki

- Skenaariossa ei kuvattu järjestelmän käyttöliittymän yksityiskohtia
  - Käyttöliittymää koskevat suunnittelupäätökset on tällä tasolla syytä jättää avoimiksi selkeyden vuoksi
  - Toiminnallisissa skenaarioissa ei ole toisaalta syytäkään vielä kiinnittää käyttöliittymän suunnittelua, vaan mahdollistaa monia toteutuksia (vaihtoehtoja on!)
  - Käytettävyydellä ja käyttöliittymällä on kuitenkin usein vaikutusta arkkitehtuuriin, joten niitäkin pitää projektin aikaisessa vaiheessa alkaa miettiä

29.9.2015

581385 Ohjelmistoarkkitehtuurit

18

Ylimmän tason rajapintamalli

## SYSTEMIKONTEKSTI

29.9.2015

581385 Ohjelmistoarkkitehtuurit

19

## Systemikonteksti

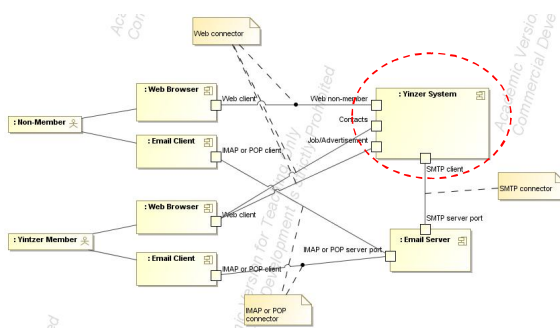
- Kuten käyttötapauskaavio, *systemikontekstikaavio* (system context diagram) näyttää SUD:n kanssa vuorovaikutuksessa olevat ulkoiset järjestelmät / toimijat
  - Mallinnetaan *komponentteina*
- Erona toiminnallisuuteen keskittyvään käyttötapauskaavioon systemikonteksti näyttää
  - järjestelmiä yhdistävät kommunikaatiokanavat eli *konnektorit*
  - *Portit*, joiden kautta kommunikaatio tapahtuu ja jotka kokoavat loogisesti yhteen kuuluvat rajapinnat samaan yhteyspisteeseen

29.9.2015

581385 Ohjelmistoarkkitehtuurit

20

## Yinzer - systemikontekstikaavio



29.9.2015

581385 Ohjelmistoarkkitehtuurit

21

## Systemikonteksti

- Systemikontekstikaavio ottaa kantaa konkreettisiin vuorovaikutusmekanismeihin
  - Konnektorit ja portit
- Käyttötapauskaavio näyttää Yinzer -jäsenen kommunikoivan suoraan järjestelmän kanssa, mutta systemikonteksti näyttää sen tapahtuvan web-selaimen välityksellä kahden teknisesti samanlaisen mutta loogisesti erillisen kanavan ja portin kautta
  - Lisäksi näytetään sähköpostiliikenteen kulku sähköpostipalvelimen ja sen asiakkaiden kautta
  - Porttien tukemat käyttötapaukset määrittävät porttien *roolit*
- Käyttötapauksen ja skenaarioiden kuvaamat vuorovaikutukset ulkoisten järjestelmien ja toimijoiden kanssa *pitää pystyä selittämään* systemikontekstin kuvaamien kommunikaatioväylien avulla
  - Jokaiselle käyttötapaukselle pitää olla sen mahdollistava portti / portit

29.9.2015

581385 Ohjelmistoarkkitehtuurit

22

## Systemikonteksti

- Huomaa, että systemikontekstikaavio näyttää *komponentti-instansseja*, eli ajonaikaisen tilanteen
  - Järjestelmätason komponentti-instansseja on usein vain yksi (esim. SUD eli Yinzer), mutta asiakaskomponentteja voi olla useita näitten eri roolien havainnollistamiseksi (Yinzer-jäsenet ja ei-jäsenet)
  - Mitä isommasta komponentista on kysymys, sitä todennäköisempää on, että siitä koskaan on olemassa vain yksi tai korkeintaan muutama instanssi järjestelmän käytössä ollessa
- Systemikontekstikaavio on vain yksi esimerkki kaikista mahdollisista ajonaikaisista konfiguraatioista
  - Asiakkaiden ja s-postipalvelimien määrä vaihtelee järjestelmän elinkaaren aikana paljonkin

29.9.2015

581385 Ohjelmistoarkkitehtuurit

23

## Systemikonteksti

- *Neuvo:*
  - Komponenttien *kaikki portit* ja niitä yhdistävät konnektorit pitäisi aina näyttää, kun komponentteja piirretään kaavioihin
    - Näin on todennäköisempää, että kaavioiden avulla todella voi ymmärtää järjestelmän toimintaa

29.9.2015

581385 Ohjelmistoarkkitehtuurit

24

Yliymän tason rajapintamalli

## MODUULIT

29.9.2015

581385 Ohjelmistoarkkitehtuurit

25

## Moduulit

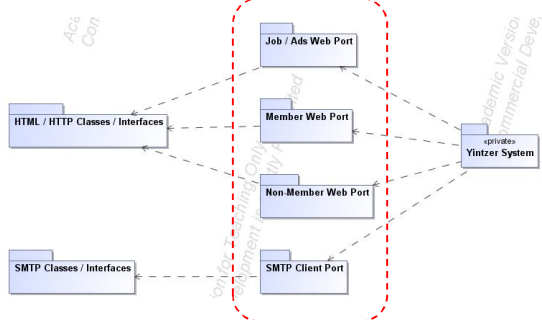
- Moduulimalli kuvaa järjestelmän *ulospäin näkyviksi tarkoitetut* koodimoduulit (pakkaukset)
  - Nämä liittyvät järjestelmän *porttien toteutukseen*
  - Moduulit sisältävät ne määrittelyt (luokat, rajapinnat, tietotyypit ja niiden käyttöön vaadittavan implementaation) joita tarvitaan porttien/rajapintojen käyttöön ulkoisista ohjelmistoista
- Yleisempänä pyrkimyksenä on noudattaa ohjelmointityyliä, joka tekee *arkkitehtuurin näkyväksi* myös implementaation tasolla (eli koodimallissa)

29.9.2015

581385 Ohjelmistoarkkitehtuurit

26

## Moduulimalli



29.9.2015

581385 Ohjelmistoarkkitehtuurit

27

Yliymän tason rajapintamalli

## SIJOTTELU SUORITUSYMPÄRISTÖÖN

29.9.2015

581385 Ohjelmistoarkkitehtuurit

28

## Sijoittelumalli

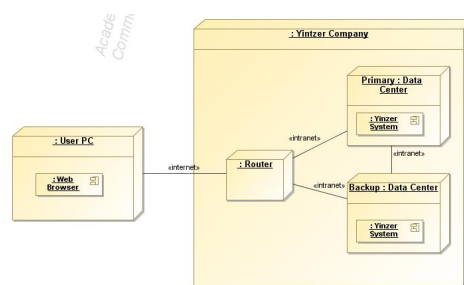
- Yinzter-järjestelmän suoritusympäristö (laitteisto, tietoliikenneyhteydet) vaikuttaa merkittävästi järjestelmän suoriutumiseen tehtävistään
- Sijoittelukaavio (deployment diagram) kuvaa, miten järjestelmän komponenttien instanssit on allokoitu suoritusympäristön laitteistoon (palvelimet, verkot, erilaiset päätelaitteet) ja mitä kommunikaatiokanavia ne käyttävät

29.9.2015

581385 Ohjelmistoarkkitehtuurit

29

## Yinzter-sijoittelukaavio



29.9.2015

581385 Ohjelmistoarkkitehtuurit

30

## Sijoittelumalli

- Ohjelmistojen asennukseen, varmistukseen, päivitykseen ja migraatioon liittyy monenlaisia teknisiä haasteita
  - Näistä kannattaa laatia toiminnallisia skenaarioita, jotka kuvaavat esimerkiksi ohjelmistoversion päivitykseen tai varmuuskopiointiin liittyvät työvaiheet ja järjestelmän tilan

29.9.2015

581385 Ohjelmistoarkkitehtuurit

31

Yliimmän tason rajapintamalli

## LAATUATTRIBUUTIT

29.9.2015

581385 Ohjelmistoarkkitehtuurit

32

## Laatuattribuutit ja arkkitehtuuriajurit

- Järjestelmän laatuominaisuudet on syytä priorisoida toimivan teknisen ratkaisun löytämiseksi
  - Priorisointi on tehtävä näkyväksi kaikille kehittäjille
- Laatuvaatimukset voidaan kuvata eksplisiittisemmin myös laatuattribuuttiskenaarioita käyttäen (quality attribute scenarios)

29.9.2015

581385 Ohjelmistoarkkitehtuurit

33

## Laatuattribuuttiskenaarioesimerkki

| Skenaario             | #<tunniste>   |
|-----------------------|---|
| Lähde                 | Yinzer-jäsen  |
| Heräte                | Web-sivun pyyntö Yinzer-palvelimelta  |
| Ympäristö             | Normaali toiminta   |
| Artefakti             | Koko järjestelmä  |
| Vastaus               | Palvelin palauttaa pyydetyn web-sivun   |
| Mittari               | Yinzer-järjestelmä lähettää vastauksen yhden (1) sekunnin kuluessa  |
| Koko skenaarion kulku | Yinzer-jäsen klikkaa linkkiä web-selaimessa; selain lähettää sivupyynnön Yinzer-palvelimelle, joka lähettää vastauksena web-sivun yhden sekunnin kuluessa |

29.9.2015

581385 Ohjelmistoarkkitehtuurit

34

## Laatuattribuuttiskenaariot

- Todella merkittäviä laatuattribuuttiskenaarioita on yleensä vain kourallinen järjestelmää kohden
  - Tulisi siis valita oikeasti tärkeimmät laatuominaisuudet ja niihin liittyvät konkreettiset vaatimukset skenaarioihin
- Laatuskenarion määrittely on helppoa, jos attribuutilla on selkeä ja suoraviivaisesti laskettava ja mitattava mittari
  - Esimerkiksi ylläpidettävyyteen liittyvien mittarien määrittely voi olla huomattavan hankalaa

29.9.2015

581385 Ohjelmistoarkkitehtuurit

35

## "Arkkitehtuuriajurit"

- Laatuattribuuttiskenaariot voidaan priorisoida<sup>1</sup> niiden suhteellisen tärkeyden ja toisaalta niiden toteuttamisen vaikeuden perusteella
- Sekä hyvin tärkeät että vaikeasti toteutettavat laatuskenaariot vaativat aivan erityistä huomiota suunnittelussa ja toteutuksessa
  - Tämän takia niitä kutsutaan *arkkitehtuuriajureiksi* (architectural driver)
  - Suunnittelupäätösten ja –valintojen kelpoisuus arvioidaan niitä vasten

<sup>1</sup>Tähän palataan kurssilla arkkitehtuurin arvioinnin yhteydessä

29.9.2015

581385 Ohjelmistoarkkitehtuurit

36

## Laatuominaisuuksien tasapainottelu

- Järjestelmän suunnittelussa jostakin laatuominaisuudesta täytyy yleensä tinkiä jonkin toisen hyväksi (trade-off)
  - Esimerkiksi Yinzer lähettää ei-jäsenille sähköpostitse kontaktiksi -kutsun yhteydessä linkin sivulle, joka sisältää joitakin henkilökohtaisia tietoja kutsujasta
  - Sähköpostin saaja voi klikata linkkiä ja nähdä sivun suoraan – sivu ei vaadi lataajan tunnistamista. Toisaalta linkin URI:iin sisältyy suuri satunnaisluku, jonka arvaaminen ilman linkkiä on aika epätodennäköistä.
  - Turvallisemmat ratkaisut, joissa klinkin likkaaja jollain tavoilla autentikoitaisiin, vaikuttaisivat heikentävästi käytettävyyteen lisäämällä uusia askeleita käyttötapukseen. Tämä voisi vähentää myös potentiaalisen jäsenen halua tulla mukaan. Nyt käyttäjä näkee hyvin nopeasti, mistä kutsussa on oikein kysymys ja voi päättää, haluaako mukaan palveluun vai ei.

29.9.2015

581385 Ohjelmistoarkkitehtuurit

37

## Rajapintamallin yhteenveto

- Suunnitelumallin ylimmän tason rajapintamalli (boundary model) sisältää siis seuraavat osat jotka kuvaavat *järjestelmän ulospäin näkyviä piirteitä*
  - Käyttötapaukset ja toiminnalliset skenaariot
  - Systeemikonteksti
  - Portteihin liittyvät moduulit
  - Sijoittelu laitteistoon
  - Laatuominaisuudet ja laatuskenaariot

29.9.2015

581385 Ohjelmistoarkkitehtuurit

38

## SISÄRAKENNEMALLI

29.9.2015

581385 Ohjelmistoarkkitehtuurit

39

## Sisärakennemalli

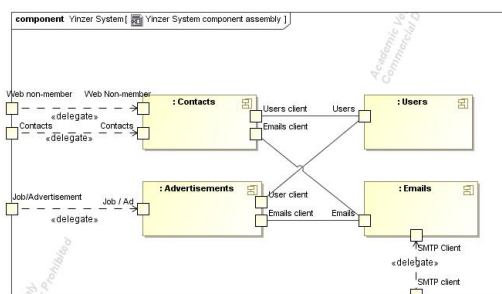
- Komponenttikokoonpano** (Component Assembly) näyttää joukon komponentti-instansseja ja niitä porttien kautta yhdistävät konektorit
  - Yinzer –systeemikontekstikaavio on esimerkki tällaisesta konfiguraatiosta
- Sisärakennemalleissa** komponenttikokoonpanoilla määritellään rajapintamalleissa esiteltujen komponenttien sisäinen suunnittelu, joka piilotettiin rajapintamalleissa

29.9.2015

581385 Ohjelmistoarkkitehtuurit

40

## Yinzer –systeemin ylimmän tason sisärakenne



29.9.2015

581385 Ohjelmistoarkkitehtuurit

41

## Yinzer –systeemin sisärakenne

- Sisärakennemallissa näytetään kaikki rajapintamallissa määritellyt portit sekä komponentit, jotka loogisesti ovat systeemi-komponentin sisällä ja muodostavat sen kokoonpanon
- Kokoonpano näyttää, kuinka Yinzer-systeemin portit delegoivat toimintansa sisärakenteen komponenttien portteihin
  - Ulompi portti ei itse asiassa tee mitään, vaan delegaatti hoitaa kaiken työn
  - Porttien on oltava vähintään yhteensopivia, joskin sisempi portti voi tarjota enemmän operaatioita, jotka eivät ole näkyvissä ulommassa portissa

29.9.2015

581385 Ohjelmistoarkkitehtuurit

42

## Sisäkkäiset rakenteet

- Rajapinta- ja sisärakennemallit muodostavat rekursiivisen rakenteen
  - Yinzer-järjestelmän rajapintamallin systeemikontekstikaavio näytti Yinzer-systeemin yhtenä komponenttina, joka oli porttiansa ja niihin kytkettyjen konektorien kautta yhteydessä ulkoisiin toimijoihin
  - Ylimmän tason sisärakennemalli näyttää systeemikomponentin välittömän komponenttikokoonpanon
  - Tämä kaavio toimii samalla rajapintamallina kokoonpanon komponenteille
    - Niillä voi olla kullakin oma sisärakennemallinsa, joka näyttää rekursion seuraavan tason kokoonpanon

29.9.2015

581385 Ohjelmistoarkkitehtuurit

43

## Sisäkkäiset rakenteet

- Sisärakenteen tarkentamista (refinement) kannattaa jatkaa vain tiettyyn rajaan asti (itse päätettävä)
  - Jollekin tarkennustasolle tultaessa on yleensä jo järkevämpää siirtyä suoraan varsinaiseen implementaatioon (koodiin)

29.9.2015

581385 Ohjelmistoarkkitehtuurit

44

Sisärakennemalli

## TOIMINNALLISET SKENAARIOT

29.9.2015

581385 Ohjelmistoarkkitehtuurit

45

## Toiminnalliset skenaariot, osa II

- Toiminnallisia skenaarioita voi käyttää sisärakennemallissa kuten rajapintamallissakin toiminnallisuuden kuvaamiseen
  - Nyt skenaarion askeleisiin voi kuitenkin sisällyttää kuvauksen, mitä *järjestelmän sisäiset komponentin tekevät* niiden yhteydessä

29.9.2015

581385 Ohjelmistoarkkitehtuurit

46

## Toiminnalliset skenaariot, osa II

| Nimi     | Kevin saa toita Widgetron-yhtiöstä   |
|----------|--|
| Alkutila | Alan ja Owen ovat jäseniä Yinzerissä; Kevin ei ole jäsen. Alan on toissa Widgetronissa   |
| Toimijat | Alan, Kevin, Owen, Widgetron   |
| Askelet  | <ol style="list-style-type: none"> <li>1. Alan kutsuu Kevinin kontaktikseen (Contact) omaan verkostoonsa (Network). / Järjestelmä lähettää Kevinille kutsun sähköpostissa.           <ol style="list-style-type: none"> <li>a) Contacts-komponentti etsii Kevinin sähköpostiosoitetta Users-komponentin palveluja käyttäen. Kevinia ei löydy Yinzer-käyttäjien joukosta.</li> <li>b) Contacts-komponentti muodostaa sähköpostiviestin Kevinille kutsuen hänet jäseneksi Yinzeriin ja Alanin verkostoon. Contacts pyytää Email-komponenttia lähettämään viestin</li> </ol> </li> <li>2. Kevin klikkaa sähköpostiviestissä olevaa linkkiä, liittyy jäseneksi Yinzeriin ja hyväksyy Alanin kutsun kontaktipyynnön.</li> <li>...</li> <li>...</li> </ol> |

Sisärakennemalli

## KOMPONENTTIEN VASTUUT, SUUNNITTELURAJOITTEET

29.9.2015

581385 Ohjelmistoarkkitehtuurit

48



## Arkkitehtuuri-elementtien vastuut

- Mallit käyttävät lyhyitä nimiä kuvaamaan kompleksisia asioita (esim. komponentin tai portin nimi)
- On kuitenkin erittäin tärkeää ymmärtää elementtien *vastuut*<sup>1</sup> (responsibility) väärinkäsitysten ja väärin oletusten välttämiseksi
  - Mitä komponentti *tietää*?
  - Mitä komponentti *tekee*?
  - Mikä on sen *rooli* – mikä ei toimisi järjestelmässä tai jäisi tekemättä, jos komponenttia ei olisi?
  - Mikä on sen *salaisuus*?
- Vastuiden kirjaaminen on helppo ja halpa tapa lisätä mallin ymmärrettävyyttä ja käyttökelpoisuutta

<sup>1</sup>Katso esim. Responsibility-driven design, CRC -cards

## Rajoitteiden ohjausvaikutus

- *Rajoitteiden* (constraints) tarkoitus on ohjata (guide-rails) suunnittelua ja toteutusta tunnistettujen riskien minimoimiseksi ja laatuvaatimusten saavuttamiseksi
- Esimerkiksi
  - Pyritään rajoittamaan mitä käyttöjärjestelmän palveluja ohjelma saa käyttää, jotta sen siirtäminen eri käyttöjärjestelmän päälle olisi helpompaa
  - Asetetaan komponenttien RAM-muistin käytölle yläraja, jotta ohjelma voi toimia sulautetussa järjestelmässä, jossa on vain vähän muistia käytössä

## Rajoitteet

- Yinzer-järjestelmän suunnittelussa laatuskenaario antaa tiukan rajoitteen web-sivupyynnön vastausajalle ( $\leq 1s$ )
  - Yksi osa ratkaisua on vaatia asynkronisen konektorin käyttöä web-pyyntöjä käsittelevien komponenttien ja Email-komponentin välillä, koska sähköpostin lähettäminen saattaa kestää arvaamattoman pitkään
- Rajoite on kehittäjän *ystävä*, ei riesa
  - Rajoitteet on syytä dokumentoida ja automatisoida niiden valvonta (jos vain mahdollista)

## Yhteenveto

- Sisärakennemalli
  - Sisäkkäisten rakennemallien muodostama puumainen hierarkia, jonka juurena on koko järjestelmäkomponentin sisärakenne komponenttikokoonpanona ja alahaaroina näitten komponenttien rakennemallit
  - Rajapintamallin toiminnallisia skenaarioita voidaan tarkentaa komponenttien toimintojen ja yhteistyön kuvauksella
  - Komponenttien vastuut kannattaa selittää
  - Rajoitteet ohjaavat suunnittelua ja toteutusta laatuvaatimusten saavuttamiseksi ja riskien minimoimiseksi

## Yinzer –suunnittelumallin rakenne

- Ylimmän tason rajapintamalli
  - Käyttötapaukset ja toiminnalliset skenaariot
  - Systeemikonteksti
  - ulos näkyvät koodimoduulit
  - sijoittelu laitteistoon
  - laatuskenaariot
- Ylimmän tason sisärakennemalli
  - Komponenttikokoonpano
  - Tarkennetut toiminnalliset skenaariot, jotka selostavat komponenttien yhteistoiminnan
  - Contacts -komponentin malli
    - Komponenttikokoonpano
    - Toiminnalliset skenaariot tarkennettuna tälle tasolle
  - .... (muitten komponenttien mallit)

## Katso myös

- Kurssikirjan 4. luvussa on yksityiskohtaisempi esimerkki arkkitehtuurimallien käytöstä osana riskien hallinnan perustalta lähtevää arkkitehtuurityötä (risk-driven approach)