Suunnittelumalli
Design Model

Luento 10

Oppimistavoitteet

- Suunnittelumalli
  - Rajapintamalli, sisärakennemallit

SUUNNITTELUMALLI

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

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

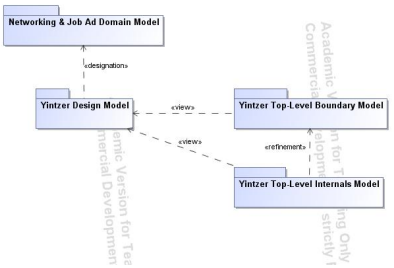
Suunnittelumalli

- Näkymä *(View)*
  - Projektio, 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

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

Yinzer Top-Level Design Model



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

YLIMMÄN TASON RAJAPINTAMALLI

Ylimmän tason rajapintamalli

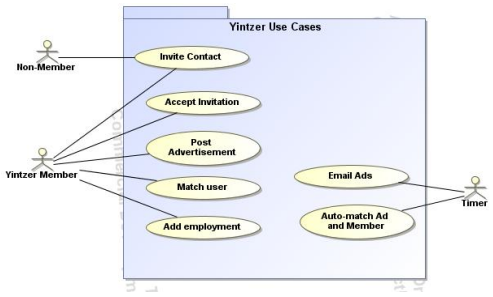
KÄYTTÖTAPAUKSET

Rajapintamalli
Käyttötapaukset ja skenaariot

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

[1]SUD = system under design, suunnitteilla oleva järjestelmä

Käyttötapauskaavio



Ylimmän tason rajapintamalli

TOIMINNALLISET SKENAARIOT

Toiminnalliset skenaariot

- *Skenaariot* kuvaavat täydellisiä tapahtumakulkuja, joissa SUD on vuorovaikutuksessa ulkopuolisten toimijoiden (actor) kanssa
  - Erona sovellusalueemallin skenaarioihin nyt voidaan jo puhua Yintzer-*järjestelmästä* ja siihen kuuluvista jäsenistä (member) erotuksena muista ihmisistä
  - Skenaarion kulussa 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

Skenaarioesimerkki

Nimi	Kevin saa töitä Widgetron-yhtiöstä
Alkutila	Alan ja Owen ovat jäseniä Yinzerissä; Kevin ei ole jäsen. Alan on töissä Widgetronissa
Toimijat	Alan, Kevin, Owen, Widgetron
Askelet	1. Alan kutsuu Kevinin kontaktikseen (Contact) omaan verkostoonsa (Network). / Järjestelmä lähettää Kevinille kutsun sähköpostissa. 2. Kevin klikkaa sähköpostiviestissä olevaa linkkiä, liittyy jäseneksi Yinzeriin ja hyväksyy Alanin kutsuun liittyvän kontaktipyynnön. 3. Widgetron julkaisee ilmoituksen (Ad) avoimesta ohjelmistokehittäjän toimesta (Job). / Järjestelmä automaattisesti ehdottaa Owenia rekrytoitavaksi ja lähettää hänelle sähköpostin. 4. Alan näkee ilmoituksen ja ehdottaa Keviniä toimeen. / Järjestelmä lähettää Kevinille sähköpostin. 5. Kevin saa työpaikan ja hän päivittää Yinzer-profiiliinsa uuden työnsä Widgetron-yhtiössä.

Skenaarioesimerkki

- Jokainen edellä olevan skenaarion erillinen askel vastaa yhtä käyttötapausmallin käyttötapausta
  - Käyttötapausmalli kuvaa mahdolliset tapaukset, mutta skenaario kuvaa jonkin tietyn ketjun käyttötapausten suorituksia
  - Skenaarioiden ei tarvitse käyttää kaikkia käyttötapauksia; 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

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 käyttöliittymän vaatimuksilla on kuitenkin usein vaikutusta arkkitehtuuriin, joten niitäkin pitää projektin aikaisessa vaiheessa alkaa miettiä
  - Käytettävyyten ja vuorovaikutukseen liittyviä asioida voidaan käsitellä esimerkiksi *laatuattribuutteina* (myöh.)

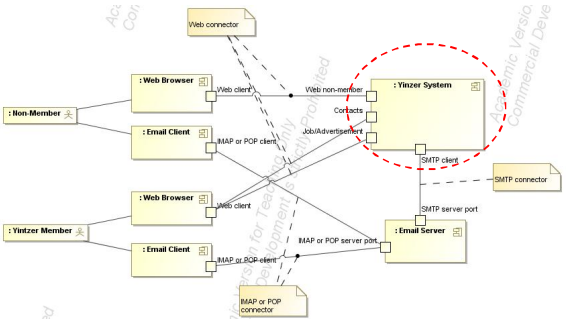
Ylimmän tason rajapintamalli

SYSTEEMIKONTEKSTI

Systeemikonteksti

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

Yinzer - systeemikontekstikaavio



Systeemikonteksti

- Systeemikontekstikaavio ottaa kantaa konkreettisiin vuorovaikutusmekanismeihin
  - Konnektorit ja portit
- Käyttötapauskaavio näyttää Yinzer -jäsenen kommunikoivan suoraan järjestelmän kanssa, mutta systeemikonteksti 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*
- Käyttötapausten ja skenaarioiden kuvaamat vuorovaikutukset ulkoisten järjestelmien ja toimijoiden kanssa *pitää pystyä selittämään* systeemikontekstin kuvaamien kommunikaatioväylien avulla
  - Jokaiselle käyttötapaukselle pitää olla sen mahdollistava portti / portit

Systeemikonteksti

- Huomaa, että systeemikontekstikaavio 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 on olemassa vain yksi tai korkeintaan muutama instanssi järjestelmän käytössä ollessa
- Systeemikontekstikaavio on vain yksi esimerkki kaikista mahdollisista ajonaikaisista konfiguraatioista
  - Asiakkaiden ja s-postipalvelimien määrä vaihtelee järjestelmän elinkaaren aikana paljonkin

Systeemikonteksti

- *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

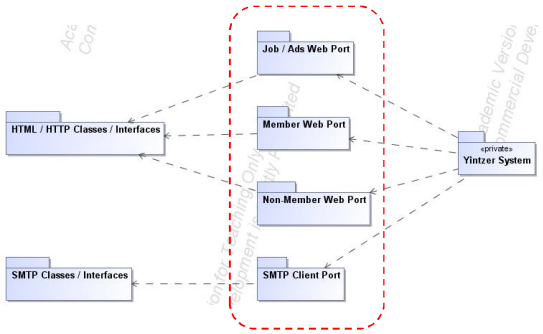
Ylimmän tason rajapintamalli

MODUULIT

Moduulit

- Moduulimalli kuvaa järjestelmän *ulospäin näkyviksi tarkoitetut* koodimoduulit (pakkaukset)
  - Nämä liittyvät järjestelmän *porttien toteutukseen*
  - Moduulit sisältävät ne määrittelyt (luokat, rajapinnat, tietotyyypit 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)

Moduulimalli



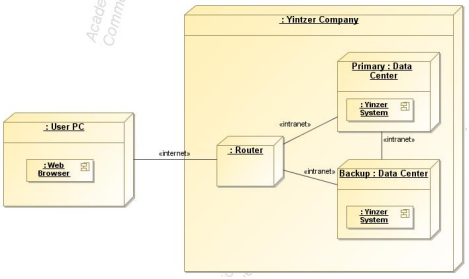
Ylimmän tason rajapintamalli

SIJOITTELU SUORITUSYMPÄRISTÖÖN

Sijoittelumalli

- Yinzer-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

Yinzer-sijoittelukaavio



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

Ylimmän tason rajapintamalli

LAATUATTRIBUUTIT

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)

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

Laatuattribuuttiskenaariot

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

”Arkkitehtuuriajurit”

- Laatuattribuuttiskenaariot voidaan priorisoida[1] 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

[1]Tähän palataan kurssilla arkkitehtuurin arvioinnin yhteydessä

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 linkin klikkaaja jollain luotettavalla tavalla tunnistettaisiin, vaikuttaisivat heikentävästi käytettävyyteen lisäämällä uusia askeleita käyttötapaukseen. 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.

Rajapintamallin yhteenveto

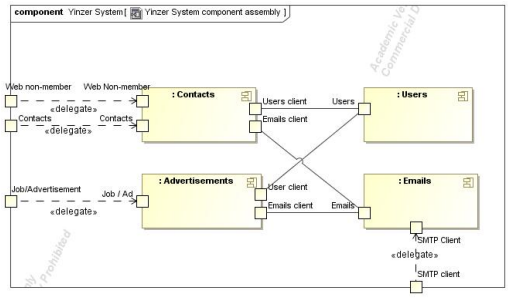
- Suunnittelumallin ylimmän tason rajapintamalli (boundary model) sisältää siis seuraavat osat jotka kuvaavat *järjestelmän ulospäin näkyviä piirteitä*
  - Käyttötapaukset ja toiminnalliset skenaariot
  - Systeemikonteksti
  - Portteihin liittyvät moduulit
  - Sijoittelu laitteistoon
  - Laatuominaisuudet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

SISÄRAKENNEMALLI

Sisärakennemalli

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

Yinzer –systeemin ylimmän tason sisärakenne



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

Sisäkkäiset rakenteet

- Rajapinta- ja sisärakennemallit muodostavat rekursiivisen rakenteen
  - Yinzer-järjestelmän rajapintamallin systeemikontekstikaavio näytti Yinzer-systeemin yhtenä komponenttina, joka oli porttiensa ja niihin kytkettyjen konnektorien 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

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)

Sisärakennemalli

TOIMINNALLISET SKENAARIOT

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

Toiminnalliset skenaariot, osa II

Nimi	Kevin saa töitä Widgetron-yhtiöstä
Alkutila	Alan ja Owen ovat jäseniä Yinzerissä; Kevin ei ole jäsen. Alan on töissä Widgetronissa
Toimijat	Alan, Kevin, Owen, Widgetron
Askelet	1. Alan kutsuu Kevinin kontaktikseen (Contact) omaan verkostoonsa (Network). / Järjestelmä lähettää Kevinille kutsun sähköpostissa. a) Contacts-komponentti etsii Kevinin sähköpostiosoitetta Users-komponentin palveluja käyttäen. Keviniä ei löydy Yinzer-käyttäjien joukosta. b) Contacts-komponentti muodostaa sähköpostiviestin Kevinille kutsuen hänet jäseneksi Yinzeriin ja Alanin verkostoon. Contacts pyytää Email-komponenttia lähettämään viestin 2. Kevin klikkaa sähköpostiviestissä olevaa linkkiä, liittyy jäseneksi Yinzeriin ja hyväksyy Alanin kutsun kontaktipyynnön. ... ...

Sisärakennemalli

KOMPONENTTIEN VASTUUT, SUUNNITTELURAJOITTEET

Arkkitehtuurielementtien vastuut

- Mallit käyttävät lyhyitä nimiä kuvaamaan kompleksisia asioita (esim. komponentin tai portin nimi)
- On kuitenkin erittäin tärkeätä ymmärtää elementtien *vastuut*[1] (resposibility) väärinkäsitysten ja väärien 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

[1]Katso esim. Reponsibility-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
  - Kielletään joitain komponentteja luomasta omia suoritussäikeitään (thread), koska k:n suoritusympäristö (kehys, containeri) huolehtii tästä

Rajoitteet

- Yinzer-järjestelmän suunnittelussa laatuskenaario antaa tiukan rajoitteen web-sivupyynnön vastausajalle (≤ 1s)
  - Yksi osa ratkaisua on vaatia asynkronisen konnektorin 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)