Ohjelmistoarkkitehtuurit

Laatutekijät arkkitehtuurisuunnittelussa

27.9.2012 1

Laatutekijät arkkitehtuurisuunnittelussa

- Ohjelmistoja kehitettäessä ***toiminnallisuuden*** suunnittelu saa usein kaikkein keskeisimmän osan
- Järjestelmän arkkitehtuurin kannalta suoraan toiminnallisuuteen liittyvät seikat ovat kuitenkin toisarvoisia
- Jos toiminnallisuus olisi ***ainoa*** järjestelmälle asetettava vaatimus, koko järjestelmä voisi olla yksi monoliittinen komponentti vailla minkäänlaista sisäistä rakennetta
- Järjestelmälle asetettavat ***laatuvaatimukset*** edellyttävät järjestelmältä (esimerkiksi) rakenteellisuutta

27.9.2012 2

Laatutekijät arkkitehtuurisuunnittelussa

- Onnistunut arkkitehtuuri on edellytys järjestelmän keskeisten laatuvaatimusten saavuttamiseksi
- ***Pelkät*** arkkitehtuuritason ratkaisut eivät kuitenkaan takaa laatuvaatimusten täyttymistä
  - Myös yksityiskohdat ja toteutustapa vaikuttavat, mutta eivät voi parantaa huonoa arkkitehtuuriratkaisua
- Onnistuneen järjestelmän edellytyksenä on sekä kokonaisuuden (~ arkkitehtuuri) että yksityiskohtien (~ toteutus) laadun varmistaminen

27.9.2012 3

Laatutekijät arkkitehtuurisuunnittelussa

- Laatuattribuutit riippuvat toisistaan
  - Esimerkiksi millä tahansa muulla laatuattribuutilla on yleensä negatiivinen vaikutus suorituskykyyn (vrt. esim. muokattavuus, turvallisuus, saatavuus, ...)
- Ei voida keskittyä laatuattribuutteihin yksittäin, vaan kokonaisuus ratkaisee

27.9.2012 4

Laatutekijät arkkitehtuurisuunnittelussa

- Laatuattribuutteja
  - accessibility, accountability, accuracy, adaptability, administrability, affordability, auditability , autonomy, availability, credibility, process capabilities, compatibility, composability , configurability, correctness, customizability, degradability, determinability, demonstrability, dependability, deployability , distributability, durability, efficiency, evolvability, extensibility, failure transparency, fault-tolerance, fidelity, flexibility, installability, Integrity, interchangeability, interoperability , learnability, maintainability , mobility, modularity,nomadicity, operability. orthogonality, portability, precision, predictability. provability. recoverability, relevance, reliability, repeatability, reproducibility, resilience, responsiveness, reusability, robustness, safety, scalability, seamlessness, self-sustainability, serviceability, securability, simplicity, stability, standards compliance, survivability, sustainability, tailorability, testability, timeliness, traceability, ubiquity, understandability, upgradability, usability

(Wikipedia)

27.9.2012 5

Laatutekijät arkkitehtuurisuunnittelussa

- Tehokkuus (Efficiency)
  - Tehokkuus on ominaisuus, joka heijastaa ohjelmiston kykyä suoriutua suorituskykyvaatimuksista minimoimalla suoritusaikaisten resurssien käytön
    - *Resurssien käytön taloudellisuus*

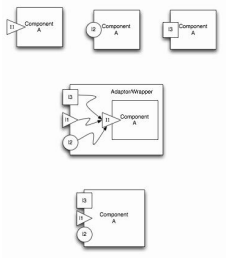
27.9.2012 6

Laatutekijät arkkitehtuurisuunnittelussa

- Tehokkuuden edistäminen arkkitehtuuritasolla
  - Pieniä komponentteja
    - *Yksi tehtävä, ylimäärän välttäminen*
  - Yksinkertaiset ja kompaktit rajapinnat
    - *Monimutkaiset rajapinnat voivat aiheuttaa tehokkuushäviöitä esim. konversiotarpeiden takia*
    - *Liian yksinkertaisetkin voivat aiheuttaa tehokkuushäviötä esim. konversiotarpeiden takia* ☺

27.9.2012 7

- Useita rajapintoja samaan toiminnallisuuteen – tehokkaampaa kuin sovittimien käyttö



27.9.2012 8

Laatutekijät arkkitehtuurisuunnittelussa

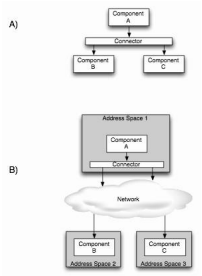
- Data- ja prosessointikomponentit erillään
- Data ja metadata erillään (pienentää kuormaa)

- Tehokkuuden edistäminen arkkitehtuuritasolla (konnektorit)
  - Valitse liitäntätavat huolellisesti
    - *Yhteistyön hoitamistapa vaikuttaa oleellisesti*
  - Tiedota vain asianosaisille
  - Suosi asynkronisia kytkentöjä – vältä odotusta

27.9.2012 9

Laatutekijät arkkitehtuurisuunnittelussa

•Käytä harkiten hajautuksen näkymättömyyttä
•Tunnista etäkohteet ja paikalliset kohteet



27.9.2012 10

Laatutekijät arkkitehtuurisuunnittelussa

- Tehokkuuden edistäminen arkkitehtuuritasolla (konfiguraatio)
  - Pidä runsaasti vuorovaikutuksessa olevat komponentit lähellä toisiaan
    - *Paikallinen data vs etädata*
    - *Paikallinen palvelu vs. etäpalvelu*
    - *Kerrosrakenteessa naapurikerros vs kaukainen kerros -> matala kerrosrakenne*
    - *Kaukaiset lähelle (cache, prefetch) – huom. ristiriita ylimäärän välttämisen kanssa*
  - Valitse liitäntätapojen käyttötilanteet huolellisesti
    - *Milloin ja missä*
  - Selvitä arkkitehtonisen tyylin tai ratkaisumallin soveltuvuus käyttötilanteeseen

27.9.2012 11

Laatutekijät arkkitehtuurisuunnittelussa

- Bass et al (1) ovat määritelleet taktiikkoja laatuominaisuuksien saavuttamiseen
- Tarkastellaan esimerkkeinä suorituskykyä ja ylläpidettävyyttä

(http://etutorials.org/Programming/Software+architecture+in+practice,+second+edition/Part+Two+Creating+an+Architecture/Chapter+5.+Achieving+Qualities/

27.9.2 12

Laatutekijät arkkitehtuurisuunnittelussa

- Suorituskykytaktiikoita
  - Suorituskykytaktiikoiden tavoitteena on tuottaa on tuottaa vastauksen johonkin tapahtumaan tietyn aikarajan puitteissa
  - Vastausaika riippuu
    1. resurssien käytöstä
    2. odotusajasta
    - *kilpailu resursseista*
    - *resurssin saatavuus*
    - *riippuvuus muusta toiminnasta*
  - Resurssitarpeeseen vaikuttavat taktiikat
    - *laskennan tehostaminen (algoritmi, välitulosten käsittely)*
    - *yleisrasitteen vähentäminen (esim. etäkutsut paikallisiksi)*
    - *tapahtumamäärän vähentäminen*
    - *tapahtumien vastaanoton kontrollointi*
    - *käsittelyajan rajoittaminen*
    - *jonojen koon rajoittaminen*

27.9.2012 13

Laatutekijät arkkitehtuurisuunnittelussa

- Suorituskykytaktiikoita
  - Resurssien hallintataktiikat
    - *Rinnakkaisuuden lisääminen*
    - *Toiminnan / datan monistaminen*
    - *Resurssien lisääminen*
  - Resurssien allokointitaktiikat
    - *jono*
    - *prioriteettijono*
    - *dynaaminen prioriteettijono*
    - *staattinen allokointi*

27.9.2012 14

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat (modifiability tactics)
  - Ohjelmiston muunneltavuutta mitataan laskemalla aikaa ja kustannuksia, jotka kuluvat ohjelmistomuutosten toteuttamiseen, testaamiseen ja käyttöönottoon
  - Muunneltavuutta lähellä olevia tekijöitä: adaptability, evolvability, maintainability
- Muunneltavuustaktiikat voidaan jakaa kolmeen ryhmään
  - Muutosten lokalisointiin perustuvat taktiikat
  - Heijastusvaikutusten (ripple effect) estämiseen perustuvat taktiikat
  - Sidonnan viivästämiseen perustuvat taktiikat

27.9.2012 15

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, muutosten lokalisointi
  - Tavoitteena rajoittaa odotettavissa olevien muutosten vaikutus mahdollisimman pieneen joukkoon komponentteja
    - *Määrätään komponenttien vastuut siten, että odotettavissa olevat muutokset koskevat rajattua joukkoa komponentteja*
  - **Lokalisointitaktiikkoja**:
    - Komponentin vastuiden semanttinen yhtenäisyys (semantic coherence)
      - *Vastuut hoidettavissa ilman laajaa ulkopuolista kommunikointia*
    - *Yleiskäyttöisten palvelujen eristäminen omiin moduuleihin*
    - Odotettavissa oleviin muutoksiin varautuminen
    - Yleistäminen
      - *Erikoistaminen parametrien kautta – parametrien tulkinta*
    - Vaihtoehtojen rajoittaminen

27.9.2012 16

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, heijastusvaikutusten esto
  - Ohjelmamuutoksen ***heijastusvaikutukset*** (ripple effects) tarkoittavat muutoksia, jotka täytyy tehdä moduuleihin, joita varsinainen muutos ei suoraan koske
  - Jos moduulia A muutetaan, niin moduulia B joudutaan muuttamaan vain siksi, että A:ta on muutettu – ei siksi että B:n ominaisuuksia on ollut tarve muuttaa
  - Tämä heijastusvaikutus johtuu siitä, että B:n toteutuksessa on jokin ***riippuvuus*** A:n toteutukseen

27.9.2012 17

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, heijastusvaikutusten esto
  - Moduulien välisiä riippuvuustyyppejä:
    - **Syntaktinen** *(muotoa koskeva)* **riippuvuus** *koskien dataa tai palveluja*
      - *Yhteinen formaatti datalle, palveluiden kutsumuodot*
    - **Semanttinen** *(merkitystä koskeva)* **riippuvuus** *koskien dataa tai palveluja*
      - *Datan/palvelun käyttäjän on tulkittava merkitys samoin kuin tuottajan*
    - **Järjestysriippuvuus** *koskien dataa tai kontrollia*
      - *Datan saapuminen tietyssä järjestyksessä*
      - *Palveluiden suorittaminen tietyssä järjestyksessä , esim. A.n pitää olla suorittanut palvelu A1 ennen kuin B voi suorittaa palvelun B1*
    - **Rajapinnan identiteetti** *tunnettava*
      - *Nimi tai kahva (”handle”) oltava tiedossa käännös ja suoritusaikana*

27.9.2012 18

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, heijastusvaikutusten esto
  - Moduulien välisiä riippuvuustyyppejä
    - Suoritusaikainen sijainti tunnettava
      - *Esim. IP-osoitteen tietäminen*
    - Datan tai palvelun laatutaso tunnettava
      - esim. tarkkuus (onko sentit mukana?)
    - Komponentin olemassaolon edellyttäminen
      - Ei voida luoda dynaamisesti
    - Moduulin käyttämien resurssien tunteminen

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, heijastusvaikutusten esto
  - Semanttisiin riippuvuuksiin perustuvia heijastusvaikutuksia on yleisesti ottaen vaikea estää
  - Voi olla epätarkoituksenmukaista tarjota kaikille käyttäjille semantiikaltaan täsmälleen samanlaista rajapintaa ja käyttäytymistä kuin ennen muutoksia
    - *Vanhan implementaation muuttamiseen on yleensä painavia syitä*
    - *Kahta erillistä implementaatiota voidaan joutua ylläpitämään siirtymäkauden ajan (vanhaa uuden rinnalla)*
  - Seuraavassa esitetyistä taktiikoista mikään ei välttämättä estä semanttisiin riippuvuuksiin perustuvia heijastusvaikutuksia

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, heijastusvaikutusten esto
  - Tiedon piilotus (Information hiding)
    - *Vanhin tekniikka muutosten propagoitumisen estämiseksi*
    - *Jaetaan komponentin vastuut pieniin osiin → piilotetaan ne osat, joiden ei tarvitse näkyä ulospäin*
    - *Jos muutos koskee piilotettua osaa, vaikutukset ulospäin ovat rajoitettuja*
    - *Liittyy odotettavissa oleviin muutoksiin varautumiseen*
      - *Muuttuvien ominaisuuksien piilottaminen on moduulijaon yhtenä pääperiaatteena*

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, heijastusvaikutusten esto
  - Olemassa olevien rajapintojen säilytys (*Maintain existing interfaces*)
    - *Jos B riippuu A:n tarjoaman rajapinnan nimestä tai kutsumuodoista, rajapinnan ja syntaksin säilyttäminen tarkoittaa, ettei B:tä tarvitse muuttaa A:n päivityksen yhteydessä*
      - *Ei välttämättä auta, jos riippuvuudet ovat semanttisia tai palveluiden/datan laatutasoon tai resurssien käyttöön liittyviä*
    - *Huom! Yleisesti rajapinnan erottaminen toteutuksesta parantaa rajapintojen vakautta*
      - *Käyttäjä riippuu vain rajapinnan (abstraktista) määrittelystä, eikä rajapinnan mistään nimenomaisesta toteutuksesta*

27.9.2012 22

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, heijastusvaikutusten esto
  - Olemassa olevien rajapintojen säilytys tehtävissä
    - *Rajapintoja lisäämällä*
      - *Monet ohjelmointikielet sallivat useita rajapintoja yhdelle ohjelmayksikölle*
      - *Uudet palvelut voidaan julkaista uuden rajapinnan kautta, jolloin vain niitä käyttäjiä tarvitsee muuttaa, jotka haluavat uusia palveluja käyttää*
    - *Sovitinta (adapter) käyttämällä*
      - *Lisätään sovitin A:n ja niiden käyttäjien välille joita ei haluta muuttaa*
      - *Sovitin tarjoaa vanhan rajapinnan käyttäen itse A:n uutta rajapintaa*
    - *Käyttämällä edustajaoliota (proxy) tynkäimplementaationa (stub)*

27.9.2012 23

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, heijastusvaikutusten esto
  - Kommunikaatioväylien rajoittaminen
    - *Rajoitetaan niiden moduulien määrää, joiden kanssa tietty moduuli käyttää samaa dataa*
    - *Rajoitetaan niiden moduulien määrää, jotka tuottavat tietyn moduulin käyttämää dataa tai jotka käyttävät sen tuottamaa dataa*
    - *Tämä vähentää datan tuotto- ja käyttösuhteista aiheutuvia heijastusvaikutuksia*
    - *Voi vaikuttaa vastuiden allokointiin moduuleille*
    - *Vaarana tehtävien kasaantuminen joillekin moduuleille*

27.9.2012 24

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, heijastusvaikutusten esto
  - Välittäjän käyttö
    - *Jos B:n riippuvuus A:sta ei ole semanttista tyyppiä, on aina mahdollista lisätä* **välittäjä** *B:n ja A:n väliin*
      - *Välittäjä piilottaa riippuvuuden konkreettisen ilmentymän ja tarjoaa B:lle muuttumattoman näkymän A:han*
      - *Välittäjä vastuulla on vain riippuvuuden hallinta, ei mitään muuta*
      - *Välittäjän käyttö on mielekästä varsinkin, jos B:n tapaisia käyttäjiä on useita tai riippuvuus on monessa kohdassa B:n toteutuksessa → A:n implementaation muuttuessa vain välittäjää tarvitsee muuttaa*

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, heijastusvaikutusten esto
  - Välittäjän käyttö
    - *Eri tyyppiset välittäjät hoitavat erilaisia riippuvuuksia:*
      - **Nimipalvelin** *mahdollistaa A:n fyysisen sijainnin (osoitteen) muuttamisen ilman vaikutuksia B:hen*
        - B käyttää A:n nimipalvelimelle tallettamaa nimeä pyytäessään nimipalvelimelta A:n osoitetta
      - **Resurssimanageri** *on välittäjä, joka on vastuussa resurssien allokoinnista järjestelmässä*
        - Resurssimanageri pyrkii täyttämään kaikkien käyttäjien tarpeet mahdollisuuksien mukaan jakamalla resursseja joidenkin ennalta määrättyjen politiikoiden ja sääntöjen mukaan

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, sidonnan viivästäminen
  - Edellä kuvatut taktiikat pyrkivät vähentämään muutosten kustannuksia pienentämällä muutettavien moduulien määrää
  - Niistä ei kuitenkaan ole hyötyä, jos halutaan lisätä ohjelmiston muunneltavuutta koskien moduulien käyttöönottotapaa ja –aikaa tai halutaan mahdollistaa muiden kuin kehittäjien tekemän muutokset
  - ***Sidonnan viivästäminen*** tukee molempia tavoitteita
    - *Nopeuttaa käyttöönottoa: järjestelmää ei tarvitse ajaa alas (ja käynnistää uudelleen)*
    - *Vaatii lisäinfrastruktuuria → kasvattaa kustannuksia*

Laatutekijät arkkitehtuurisuunnittelussa

- Muunneltavuustaktiikat, sidonnan viivästäminen
  - *Ajonaikainen rekisteröinti (runtime registration) mahdollistaa* plug-and-play operaatiot. Rekisteröinti aiheuttaa lisäkustannuksia. Tuottaja/kuluttaja rekisteröinti voidaan tehdä ajoaikaisesti tai latauksen yhteydessä.
  - *Konfigurointitiedostoissa voidaan säädellä käynnistyksen yhteydessä tehtäviä sidontoja ja parametreja.*
  - *Polymorphismi mahdollistaa metodikutsujen myöhäisen sidonnan.*
  - *Komponentin korvaus mahdollistaa latausaikaisen sidonnan.*
  - *Protokollien noudattaminen mahdollistaa prosessien ajonaikaisen sidonnan.*

27.9.2012 28

Laatutekijät arkkitehtuurisuunnittelussa

- Yksinkertaisuuden (simlicity) voidaan katsoa olevan laatutekijä, joka liittyy osatekijänä moneen muuhun laatutekijään kuten muunneltavuuteen, ylläpidettävyyteen, ymmärrettävyyteen, jne
- Vastakohtana kompleksisuus:
  - **Complexity** is a software system's a property that is directly proportional to the size of the system, number of its constituent elements, their internal structure, and the number and nature of their interdependencies

27.9.2012 29

Laatutekijät arkkitehtuurisuunnittelussa

- Kompleksisuuden vähentäminen
  - Eri tehtävät eri komponenteille (Separate concerns into different components)
    - *Kuitenkin ohjelmisto, jossa on enemmän komponentteja on kompleksisempi kuin ohjelmisto, jossa on vähemmän komponentteja*
      - *Komponenttien määrää merkittävämpi kompleksisuustekijä on eri tyyppisten komponenttien määrä*
        - Komponentin tyyppi pitää tulkita laveasti: rakenne, käyttäytyminen, kehittäjäorganisaatio, API, tekniikka aiheuttavat erityyppisyyttä. Esimerkiksi monia eri tekniikoita käyttävä ohjelmisto on kompleksisempi kuin yhteen tekniikkaan perustuva.
      - *Komponenttien määrän vähentäminen ei välttämättä vähennä ohjelmiston kompleksisuutta , jos se johtaa yksittäisten komponenttien suurempaan kompleksisuuteen.*
      - *Kaikki on suhteellista*
  - Toiminnallisuus komponentteihin – vuorovaikutus konnektoreihin
  - Komponentit semanttisesti yhtenäisiksi
  - Valmiskomponenttien käyttö voi monimutkaistaa ohjelmistoa - tiedostettava
  - Laskentakomponentit erikseen tietoformaattimuunnoksista

27.9.2012 30

Laatutekijät arkkitehtuurisuunnittelussa

- Kompleksisuuden vähentäminen
  - Konnektorit eksplisiittisesti näkyville
  - Konnektoreihin vain vuorovaikutustehtäviä
  - Eri vuorovaikutustehtävät eri konnektoreille
  - Rajoita vuorovaikutustapoja ja määriä
    - *Jälleen vuorovaikutusmekanismien määrä on merkittävä kompleksisuutta lisäävä tekijä*
    - *Yksinkertaiset vuorovaikutusmekanismit ellei ole hyvää syytä käyttää monimutkaisempia*
  - Tarpeettomien riippuvuuksien eliminointi
  - Eksplisiittiset riippuvuudet
  - Hierarkkinen osiin jako

27.9.2012 31