Metropolian tietojärjestelmäarkkitehtuuri

Nykytilan selvitys & esitys tulevaisuuden arkkitehtuurista

8.9.2009 - 2.11.2009

Jaakko Rannila, projektipäällikkö, Metropolia ammattikorkeakoulu
Eero Manninen, Java-arkkitehti, Eduix Oy
Tuukka Puumala, ohjelmoija, Eduix Oy

Termit

Seuraavassa taulussa kerrottu raportissa käytettävät käsitteet

Termi	Selite
Palvelu	Joukko kiinteästi yhteenliittyviä toimintoja, esim. opintosuunnitelman hakupalvelu
Palveluväylä	Tietojärjestelmä joka liittää palvelut yhteen ja jonka avulla palveluita voidaan koostaa uusiksi palveluiksi
Palvelualusta	Palveluväylä ja palveluiden toteuttamiseen liittyvä tekniikka
Järjestelmä	Palvelualusta ja siihen kiinteästi liittyvät muut tietojärjestelmät

Seuraavassa taulussa kerrottu raportissa käytettävät lyhenteet

Termi	Selite
ESB	Enterprice Service Bus, palveluväylä
JBI	Java Business integration, JSR 208 ja JSR 312-spesifikaatioissa kuvattu standard helpottamaan palveluiden ja tietojärjestelmien integrointia
SE	JBI-spesifikaatiossa mainittu Service Engine -komponentti, joka voidaan liittää palveluväylään tietoa prosessoivaksi komponentiksi, esim. BPEL-komponentti, XSLT-komponentti
BC	JBI-spesifikaatiossa mainittu Binding Component -komponentti, jonka avulla palveluväylä liitetään tietyn protokollan avulla rajapintaan
BPEL	Business Process Execution Language, liiketoimintaprosessien kuvaamiseen tarkoitettu XML-pohjainen kieli
SCA	Service Component Architecture, palveluiden toteuttamiseen ja koostamiseen tarkoitettu suhteellisen nuori standardi
OSGi	Standardi modulaarista alustaa varten
WS	Web Service, verkossa oleva palvelun tarjoama rajapinta
SOAP	XML-pohjainen kieli ja protokolla
EJB	Enterprise Service Bean, transaktionaalinen komponenttipohjainen tekniikka palveluiden toteuttamiseen

Miksi nykyinen arkkitehtuuri muodostaa haasteita?

Alustusta

Tietojärjestelmiä on usein lähdetty kehittämään kun ilmennyt uusia tarpeita tiedon käsittelemiseksi. Usein tietojärjestelmä ja palveluja on pidetty irrallisina kokonaisuuksia ja on ajateltu, että niiden tuottamia tietoja käytetään vain kyseisessä järjestelmässä. Myöhemmässä vaiheessa on kuitenkin todettu, että palvelun tuottamia tietoja tulisi hyödyntää myös muissa paikoissa, jolloin tietoja on lähdetty siirtämään ja/tai dublikoimaan järjestelmien välillä. Järjestelmät ovat lisäksi paisuneet, koska niille on määritelty ajan saatossa lisää tarpeita ja tarpeet ja palvelut on pyritty rakentamaan olemassa oleviin järjestelmiin, jolloin järjestelmien välille saattaa syntyä päällekkäistä tietoa samasta tietoyksiköstä.

Miksi nykyinen arkkitehtuuri muodostaa haasteita?

Nykyarkkitehtuurin ylläpitäminen ja kehittäminen tulee tulevaisuudessa entistä kalliimmaksi, koska nykyisessä arkkitehtuurissa järjestelmien välille joudutaan rakentaamaan sitä enemmän yhteyksiä, mitä enemmän kokonaisuudessa on järjestelmiä. Lisäksi tällä arkkitehtuurilla ei pystytä valvomaan riittävän hyvin tiedon eheyttä järjestelmien välillä.
Kun halutaan luoda uutta palvelua joudutaan siihen luomaan entistä enemmän yhteyksiä muihin järjestelmiin. Mikäli haluttaisiin vaihtaa joku olemassa oleva järjestelmä, niin sen vaihtamisessa jouduttaisiin varautumaan pidemmällä aikavälillä entistä suurempiin kustannuksiin, koska järjestelmään on luotu aikojen saatossa paljon riippuvuuksia ja tiedon siirtoja / tiedon lukemista eri järjestelmien välille.
Riskien ja kustannusten arviointi on lisäksi erittäin haasteellista tulevaisuudessa, koska arkkitehtuuri on niin monimutkainen. Esimerkiksi työmäärien ja kustannusten arviointi uutta järjestelmää luotaessa tai vanhaan järjestelmää kehitettäessä voi käydä niin, että joko järjelmätoimittaja tai tietohallinto ei pysty ottamaan huomioon kaikkia niitä riippuvuuksia, joita on aikojen saatossa järjestelmien välille luotu.

Tiedon eheys

Tällä hetkellä tietoa joudutaan siirtämään monen eri järjestelmän välillä. Tämä aiheuttaa mm. tiedon dublikoitumista, jolloin on mahdollista, että sama
tietoyksikkö voi esiintyä monella eri tavalla riippuen mistä asiaa kysytään. Ts. nykyinen arkkitehtuuri mahdollistaa sen, että tiedot eivät ole eheitä.

Käytännön esimerkkejä ja seurauksia

Tämä aiheuttaa mm. työresurssien turhaa käyttöä. Esim. jos opetussuunnitelma esiintyy erilaisena OPS-editorissa kuin Vuosisuunnittelussa, joutuu asiaa usein selvittämään sekä koulutuspäällikkö, suunnittelija ja tietohallintoyksikkö. Nämä ongelmat ovat tälläkin hetkellä konkreettisiä ja läsnäolevia.
Tämän lisäksi mm. raporttien laatiminen erilaisista tiedoista on haasteellista tai mahdotonta, jos tieto on erilaista eri paikoissa ja ei ole tietoa siitä kumpi tiedoista on oikeaa. Lisäksi tiedon koostaminen raportiksi monimutkaistuu, koska eri asioita joudutaan kysymään eri järjestelmistä.

Tietoturva

Nykyinen arkkitehtuuri heikentää tietoturvaa, koska ei pystytä olemaan täysin varmoja siitä, mitä erilaisia tietoja järjestelmien välillä liikkuu ja tiedon liikkumista ei pystytä riittävän hyvin monitoroimaan ja selvittämään. Nykyisessä arkkitehtuurissa on lisäksi riskinä se, että kaikkia yhteyksiä ei ole suojattu samoilla tavoilla, jolloin on mahdollista, että tiettyjen yhteyksien välillä saattaa piillä tietoturvariskejä. Järjestelmien yhteydet on toteutettu erilaisilla teknologioilla, jonka vuoksi yhteyksien suojaaminen samalle tasolle on haasteellista.

Tietojen käyttö eri järjestelmissä & tietojen siirtymisen seuranta

Tällä hetkellä järjestelmät joutuvat joko siirtämään tietoja muille järjestelmille tai järjestelmille tuotetaan tietoja eri paikoissa. Tästä aiheutuu mm. se, että tietyillä hetkillä järjestelmillä ei ole käytössä kaikkia niitä tietoja mitä ne tarvitsivat.

Käytännön esimerkkejä ja seurauksia

Esimerkiksi uusi opettaja ei ole käytössä kaikissa järjestelmissä, koska tietoa ei pystytä siirtämään riittävän nopeasti. Toinen esimerkki on mm. opinnon syntyminen. Kun opinto tehdään se ei ole samantien vuosisuunnittelun tai Winhan käytettävissä, jolloin opinnosta ei voida tehdä toteutusta.

Lukujärjestyksien suunnittelu

Tällä hetkellä lukujärjestyksiä ei pystytä laatimaan riittävällä tarkkudella, koska suunnitteluvälineillä ei ole tietoa siitä, mitä ovat esimerkiksi toteutusnumerot ensi lukuvuonna. Tämä tieto syntyy vuosisuunnittelussa, mutta tietoa ei edes siirretä sähköisesti suunnitteluvälineisiin vaan tieto joudutaan kaivamaan "käsin" järjestelmästä.

Tietojärjestelmät jotka hyödyntävät koostettua tietoa (esim. EHOPS & lukujärjestyksen suunnittelu)

Järjestelmä on erittäin haasteellinen toteuttaa nykyisellä arkkitehtuurilla, koska se tarvitsee tietoja monesta eri tietolähteestä ja sen tuottamia tietoja joudutaan siirtämään moneen eri järjestelmään. Jos väline toteutetaan nykyisellä arkkitehtuurilla se lisäksi aiheuttaa 1. otsikon mukaisia ongelmia.
Hyvin monet eri järjestelmät tarvitsevat tietoja erilaisista organisaation resursseista, kuten henkilöistä, tiloista ja budjeteista/rahallisesta resurssista ja esimerkiksi kustannuspaikoista. Tiedon omistaa useimmissa tapauksissa yksi järjestelmä, mutta tietoa ei joko pystytä siirtämään muiden järjestelmien käyttöön tai tietoja joudutaan dublikoimaan järjestelmien välillä, jolloin hämärtyy käyttäjillä tieto siitä, kumpi järjestelmä tiedon omistaa ja missä järjestelmässä muutoksia tietoihin saa tehdä / voi tehdä.

Esimerkki

Jos esimerkiksi tilan tiedot muuttuvat ja/tai tiloja tulee uusia, niin mihin muutokset tehdään ja miten tiedot välitetään muiden järjestelmien /
organisaation käytettäväksi.

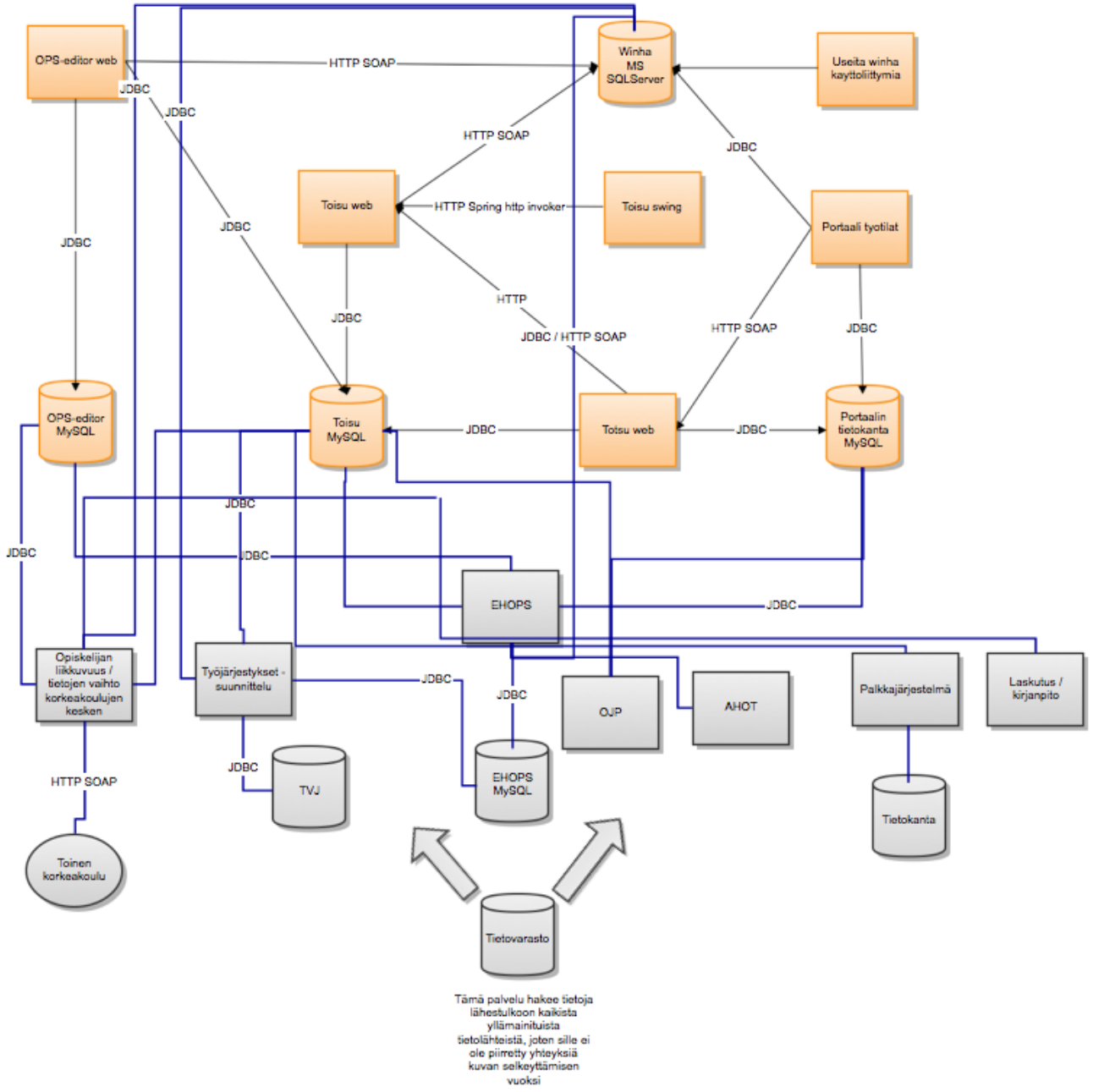
Raportointi

Nykyisellä arkkitehtuurilla joudutaan hakemaan raportointia varten tietoja monesta eri tietolähteestä ja tietoja joudutaan koostamaan. Lisäksi osa tiedoista on ristiriitaisia, koska sama tieto eri monessa muodossa eri järjestelmissä, jolloin on raportoijan vastuulla selvittää kumpi tiedoista on oikea. Tämä aiheuttaa mm. kasvanutta työresrussien käyttöä ja lisäksi pienentää raporttien luotettavuutta.

Tulevat palvelut ja niiden aiheuttamat haasteet

On odotettavissa, että tulevaisuudessa tietoja joudutaan vaihtamaan korkeakoulujen kesken sekä lisäksi erilaiset yhteistyömuodot muiden eurooppalaisten korkeakoulujen välillä kasvavat. Tällöin voi olla erittäin haasteellista vastata näihin haasteisiin nykyisellä arkkitehtuurilla, koska se ei sellaisenaan tue sitä. Sen lisäksi tulisi ottaa huomioon myös euroopassa luotavia standardeja ja mm. raketti hankkeen linjauksia. Nykyisellä arkkitehtuurilla tietojen siirtäminen tai raportointi em. esimerkeistä on erittäin hankalaa.

Kuvaus nykyisestä arkkitehtuurista



Yllä olevassa kuvassa nykyiset palvelut ja niiden väliset yhteydet on piirretty oranssilla ja

mustalla. Kuvan alalaitaan on piirretty niitä palveluita sekä palveluiden vaatimia tietolähteitä, joille on esitetty tarpeita kehittää lähitulevaisuudessa harmaalla ja sinisillä väreillä.
Kuten ylläolevasta kuvasta huomataan, on nykyinen arkkitehtuuri jo nykyisellään hyvin haastava ylläpitää ja kehittää, koska riippuvuuksia järjestelmien välillä on paljon. Tämä tarkoittaa sitä, että yhteen järjestelmään tehty muutos vaatii muutoksia myös kaikkiin niihin järjestelmiin joihin se on yhteydessä. Jos nykyisen arkkitehtuurin päälle rakennetaan niitä palveluita, joille on esitetty tarpeita ja joita mahdollisesti käynnistetään, aiheutuu siitä se, että järjestelmien väliset riiippuvuudet kasvavat entisetään. Lisäksi tulee huomioida, että uusien järjestelmien tuomat muutokset tulee ottaa huomioon myös niissä kaikissa olemassaolevissa järjestelmissä, joihin uusilla palveluilla tulisi olla yhteys.

Riskejä nykyisen arkkitehtuurilinjan jatkamisessa

- Raportoinnin vaikeus tulevaisuudessa
- Tiedon eheys jatkossa
- Riskit tietojen korpputoitumiselle
- Uusien järjestelmien integroiminen ja kehittäminen kallista
- Vanhojen järjestelmien uusiminen ja integroiminen kallista
- Tietojen koostaminen ja hyödyntäminen haasteellista
- Tietojen vaihtaminen muiden korkeakoulujen kesken haasteellista

Millainen olisi parempi arkkitehtuuri?

Mitä hyötyjä tai etuja palvelukeskeisessä arkkitehtuurissa

Mitä on palvelupohjainen arkkitehtuuri?

Lainaus Wikipediasta:

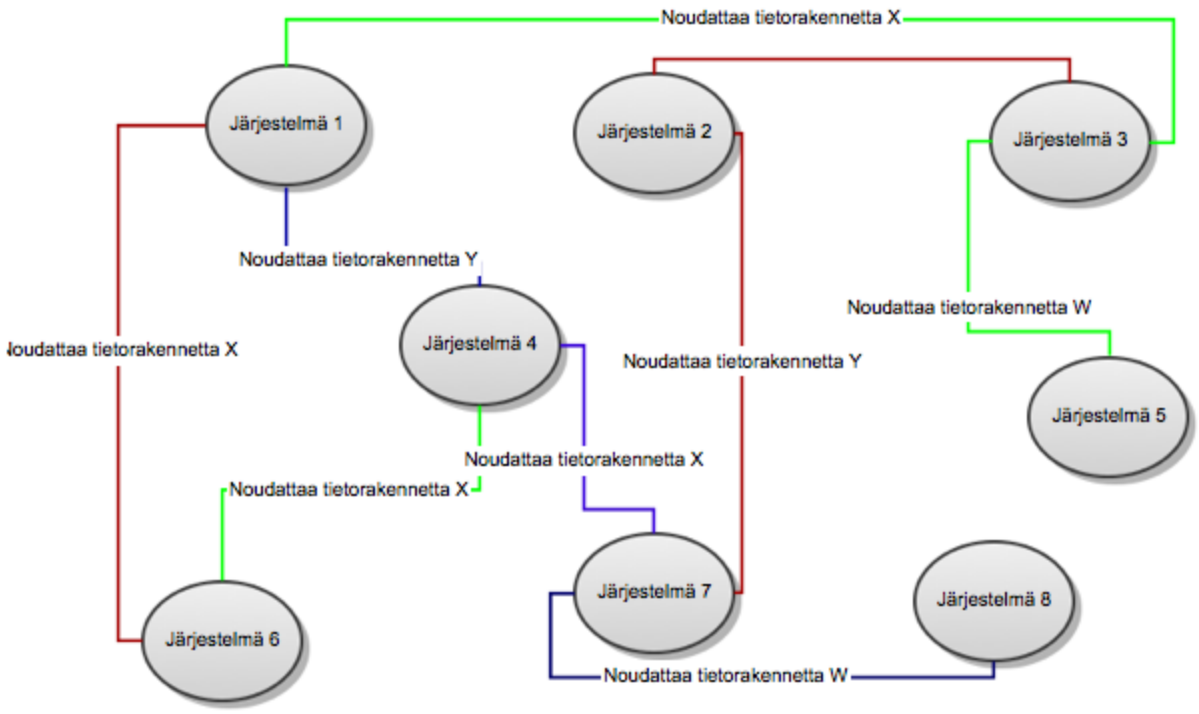
"SOA (*Service Oriented Architechture*) eli **palvelukeskeinen arkkitehtuuri** on käytetty arkkitehtuuritason suunnittelutapa, jolla eri tietojärjestelmien toiminnot ja prosessit on suunniteltu toimimaan itsenäisinä, avoimina ja joustavina palveluina. Näitä palveluita tulisi pystyä aina käyttämään avoimien standardien rajapintojen kautta. Tämän avulla pyritään aikaansaamaan erilaisten tietojärjestelmien joustava ja järjestelmäriippumaton vuorovaikutus.
SOA-arkkitehtuurin mukaan palveluita käyttävinä osapuolina toimivat esimerkiksi toiset sovellukset tai palvelut, jotka käyttävät palveluita yleensä tietoverkkojen (esimerkiksi Internetin) kautta avointen rajapintojen ja tekniikoiden (esimerkiksi Web Services) avulla. SOA-arkkitehtuurin hyödyntämisellä pyritään saavuttamaan entistä avoimempien sekä helpommin integroitavien järjestelmien toteuttaminen. SOA:n odotetaan tuovan ohjelmistojen tuottamisprosesseihin tehokkuutta vähentämällä uusien järjestelmien toteuttamiseen kuluvaa aikaa ja kustannuksia. Helpommalla integroitavuudella tavoitellaan sitä, että uudet sovellukset pystyisivät kommunikoimaan vanhojen, jo olemassa olevien sovellusten kanssa, jolloin myös tietojärjestelmien jälleenkäytettävyys voi lisääntyä.

SOA-arkkitehtuurin avulla voidaan saavuttaa etuja sekä (a) organisaatioiden olemassa olevien sisäisten, omien sovellusten välisessä vuorovaikutuksessa, että (b) ulkoisten sovellusten kanssa tapahtuvassa vuorovaikutuksessa. Tämän on tarkoitus toteutua riippumatta sovelluskehitystekniikoista (käyttöjärjestelmät, ohjelmointikieletja -tekniikat jne.). Palveluiden pitäisi olla "läpinäkyviä", siten että jos niiden sisäinen toiminta muuttuu (esimerkiksi palveluita kehitetään teknisesti tai uusia toimintoja lisätään), niiden käyttö tulisi säilyä entisellään eikä tarvetta muutoksien tekemiseen palveluita käyttäviin sovelluksiin tulisi olla.

SOA on suunniteltu ratkaisemaan samalla kertaa ainakin kaksi keskeistä ongelmaa; uusien järjestelmien nopeampi kehittäminen ja jakelu sekä eri järjestelmien välinen integrointi.
SOA eroaa perinteisestä järjestelmien keskinäisestä integroinnista siten, että SOA-mallin mukaiset järjestelmät ovat toisiinsa heikosti sidottuja, mutta silti korkeatasoiseen, usein "älykkääseen" vuorovaikutukseen keskenään pystyviä sovelluspalveluita. SOA-mallissa ajatellaan

siten jokaisen järjestelmän sisältävän palvelurajapinnan, minkä kautta sovellus voi tarjota toimintojaan "ulkopuolelle"; periaatteessa verkon kautta kaikille sallimilleen järjestelmille.
Toinen SOA:an liittyvä käsite on dynaaminen integrointi; sovellusten pitäisi pystyä automaattisesti paikantamaan tarvitsemansa palvelut, valita sopivista palveluista käyttökelpoisin ja kytkeytyä siihen automaattisesti, palvelun sisältämää rajapintaa hyväksikäyttäen."
Osana palvelukeskeistä arkkitehtuuria käytetään palvelukerrosta, jonka kautta järjestelmien välinen kommunikaatio kulkee. Tässä arkkitehtuurissa järjestelmät ovat riippuvaisia palvelukerroksesta ei monesta eri järjestelmistä.

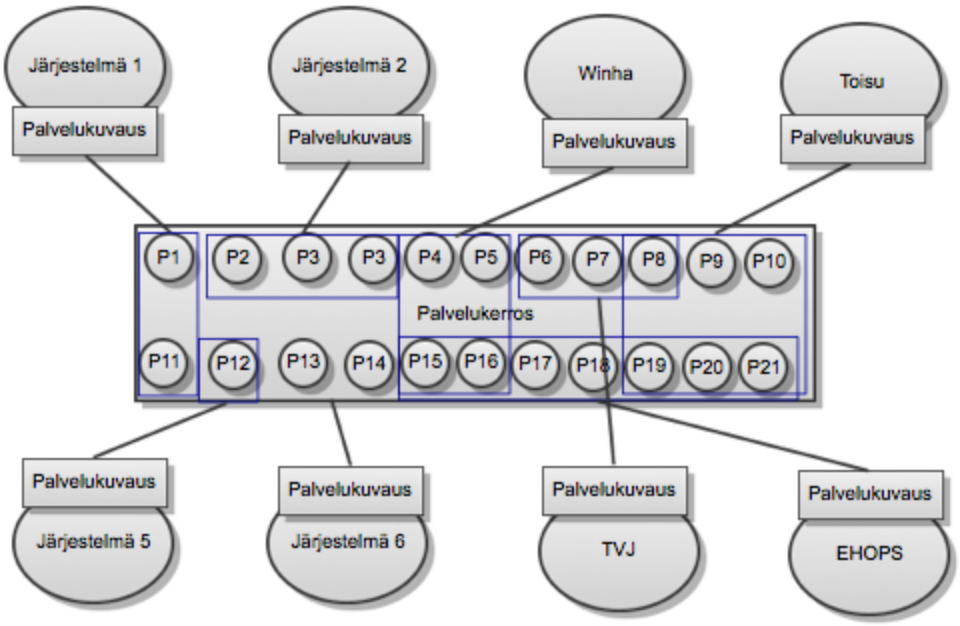
Perinteinen arkkitehtuuri



Kuten kuvasta näkyy on tietojärjestelmien välillä rakennettu yhteyksiä. Järjestelmien väliset yhteydet ovat kuitenkin toteutustavoiltaa erilaisia eli tiedon skeemaa eli rakennetta tai tyyppiä ei ole stadardisoitu ja tietoa siirretään eri tavoilla (värit).
Järjestelmät ovat tiukasti sidottuja toisiinsa, joka tarkoittaa sitä, että jos jossain järjestelmistä tulee tehdä muutoksia, niin muutokset tulee ottaa huomioon kaikissa niissä järjestelmissä, joihin järjestelmä on yhteydessä.
Jokaiseen järjestelmään jodutaan lisäksi tekemään järjestelmäintegraatioon liittyviä toimenpiteitä ja toimintoja kuten tiedon siirtymisen seurantaa ja virhelogiikkaa. Koska kaikki järjestelmät eivät kuitenkaan ole liitetty samoilla tavoilla on myös integraatioon liittyneet toiminnot erityyppisiä ja tasoisia, joka aiheuttaa mm. kasvanutta ylläpitoresurssia ja vaikeuttaa virheiden seurantaa.

Palvelukeskeinen arkkitehtuuri, jossa on hyödynnetty palvelukerrosta

Palvelukeskeissä arkkitehtuurissa pyritään ajattelemaan tiedon syntymistä ja muokkaamista eri palveluissa. Tämä eroaa aiemmasta siten, että käyttäjän ei tarvitsisi tietää misää järjestelmässä kyseistä tietoa synnytetään tai muokataan vaan enemmänkin hänen tulisi tietää miten ja missä prosesseissa tietoa muokataan ja mitä eri palveluita tähän käyttöön organisaatiossa on käytössä.
Palvelupohjainen arkkitehtuuri mahdollistaa paremmin tiedon vaihtamisen eri palveluiden välillä. Sen lisäksi palvelukeskeisen arkkitehtuurin tarkoituksen on lisätä liiketoimintakeskeistä ajattelua teknologiakeskeisen ajattelun sijasta. Ts. arkkitehtuurissa pyritään poistamaan järjestelmien teknologiaperustaisia fyysisiä rajoituksia sekä toistensa välisiä riippuvuuksia. Tämän avulla pystytään keskittymään enemmän liiketoimintaperusteiseen ajatteluun. Palvelupohjainen arkkitehtuuri mahdollistaa tiedon vaihdon ulkopuolisten toimijoiden kanssa (muut ammattikorkeakoulut, eurooppalaiset korkeakoulut). Sen lisäksi palvelukeskeisessä arkkitehtuurissa voidaan vaihtaa helpommin järjestelmätoimittajia ja tuotteita, koska palvelut on modularisoitu eikä niillä ole riippuvuuksia muiden järjestelmien kanssa.
Koska palvelukeskeissä arkkitehtuurissa pystytään vähentämään tiedon monistumista voidaan helpommin määritellä tiedon ja palveluiden (prosessien) omistajat. Tämä helpottaa mm. prosesseissa toimivien henkilöiden toimintaa, koska esim. epäselvissä tilanteissa voi suoraan olla yhteydessä prosessin / tiedon omistajaan.



Oheisessä kuvassa on kuvattu palvelukeskeinen arkkitehtuuri yksinkertaisessa muodossa.

P-merkkiset kuviot on palveluita.
Palvelukuvaus tarkoittaa sitä, että olemassa oleviin sekä uusiin järjestelmiin tuotetaan palvelukuvaus järjestelmän rinnalle, jossa kerrotaan mitä palveluita tämä järjestelmä tuottaa palvelukerrokseen.
Järjestelmät tuottavat palveluita (kuvassa P1-P21), jotka liitetään osaksi palvelukerrosta. Tarvittaessa muut järjestelmät voivat hyödyntää palvelukerrokseen tuotettuja palveluita.

Konkreettinen esimerkki (esim. EHOPS)

OPS editori tarjoaa palvelukerrokseen palvelun "anna ryhmän OPS). Winha tuottaa palvelun "Anna opiskelijan opintosuoritukset". EHOPS järjestelmä käyttää näitä palveluita ja pystyy niiden avulla koostamaan tietoa, jonka avulla se pystyy tuottamaan palvelun "Anna opiskelijan suorittamattomat pakolliset opinnot". EHOPS järjestelmän ei tarvitse tietää mikä järjestelmä tallentaa opiskelijan suoritukset tai mihin järjestelmään OPS:n tiedot on tallennettu. Tämä tieto on määritelty palvelukerrokseen ja on tarvittaessa muutettavissa, jos esimerkiksi opintosuoritusrekisteriä vaihdetaan.
Jos kyseinen palvelu haluttaisiin toteututtaa nykyisellä arkkitehtuurilla tulisi EHOPS järjestelmään joko siirtää (dublikoida) tietoa molemmista järjestelmistä tai vaihtoehtoisesti lennosta koostaa tietoa molemmista järjestelmistä, jolloin mm. suorituskykyongelmia voisi todennäköisesti esiintyä. Lisäksi nykyisessä arkkitehtuurissa järjestelmät olisivat riippuvaisia toisien järjestelmien olemassaolosta eikä ko. palvelu olisi kuin EHOPS:n käytössä, koska kyseinen palvelu on rakennettu itse järjestelmän sisään. Jos esimerkiksi joku toinen palvelu tarvitsisi samaa tietoa (yliaikaisten hakemus, AHOT jne.) tulisi palvelu rakentaa myös siihen järjestelmään (lisää kustannuksia).

Mitä ovat palvelut (käytännön esimerkki)

Saavutettavia hyötyjä palvelukeskeisessä arkkitehtuurissa

- **Parantaa prosessien hallitsemista**
- Palvelukeskeisessä arkkitehtuurissa prosessit ovat helpommin nähtävissä, koska ne määrittellään palvelukerroksen siisään eri palveluihin. Palvelupohjaisuus luo paremmat mahdollisuudet luoda prosesseja halutunlaiseksi, ilman, että järjestelmät rajoittasivat tätä liikaa. Palvelupohjainen arkkitehtuuri mahdollistaa paremmin prosessien muuttamisen, jos prosessilogiikka on rakennettu palvelukerroksen sisään. Tällöin muutokset joudutaan tekemään vain yhteen paikkaan ei kaikkiin niihin järjestelmiin, joihin prosessi liittyy.
- **Painopiste uusien järjestelmien tuottamisessa siirtyy teknologiasta palvelupohjaiseksi**
- Prosessien omistajat ja suunnittelijat "puhuvat" samaa kieltä järjestelmien toteuttajien kanssa. Järjestelmiä suunnitellessa ajatellaan enemmän palvelukokonaisuutta ja sitä mitä uutta se kokonaisuuteen tuo. Sen lisäksi huomioidaan, mitä tietoja se mahdollisesti tarvitsee muista palveluista ja prosesseista. Ts. ei lähdetä liikkelle yksittäisestä tarpeesta, jolle toteutetaan erillinen ratkaisu, vaan mietitään kokonaisuutta mihin prosessin vaiheseen tämän palvelu sijoittuisi ja miten sen tuottamia tietoja voitaisiin hyödyntää

muissa palveluissa.

- **Toteutetun palvelun uudelleenkäytettävyys**
- Kun tietty palvelu on toteutettu, sitä voidaan hyödyntää myös uusien palveluiden tuottamisessa. Tällöin tarvittava työmäärä vähenee, kun samaa järjestelmälogiikkaa ei tarvitse monistaa moniin eri tietojärjestelmiin. Tällöin myös muutokset pystytään toteuttamaan ketterämmin ja uusien palveluiden käyttöönotto yksinkertaistuu. Esimerkiksi jos tuotetaan palvelu, "hae opintojaksot opetussuunnitelmasta x" voidaan palvelua käyttää mm. Opiskelijan oppassa, opinto-oppaassa, vuosisuunnittelussa, AHOT:ssa, EHOPS:ssa jne.
- **Järjestelmäriippuvuuden väheneminen**
- Yksittäisten järjestelmien vaihtaminen helpottuu, koska riippuvuuksien hallinta yksinkertaistuu. Tällöin esimerkiksi tietojärjestelmien vaihtaminen ja kehittäminen yksinkertaistuu, koska ei jouduta sitoutumaan yksittäiseen tuotteisiin tai järjestelmäntoimittajiin. Tämä sen vuoksi, että palvelujen toteutus voidaan siirtää toiseen järjestelmään. Esimerkiksi jos Winha opiskelijarekisteri korvattaisiin RAKETTI-projektin tuottaman opiskelijarekisterillä, voitaisiin rekisteristä tuotetut palvelut siirtää käyttämään uutta järjestelmää. Tällöin ei myöskään tarvitse tehdä muutoksia muihin järjestelmiin vaan ainoastaan niihin palveluihin, joihin ko. järjestelmä toteuttaa tai tuottaa tietoja.
- **Tiedon keskittäminen ja omistajuus**
- Tieto on saatavissa yhdestä paikasta, vaikka tiedon sijainti tai omistajuus vaihtuisi. Ts. tiedot koostetaan palvelukerroksesta vaikka niiden fyysinen sijainti eroaisi eli ne sijaitsisvat eri järjestelmien ja tietokantojen alla. Tällöin esimerkiksi tietojen raportointi helpottuu. Tämän lisäksi pystytään vähentämään tiedon monistamista eri järjestelmiin, koska pyritään monistamaan ainoastaan niitä avaintietoja, jotka eivät vaihdu. Tämän takia tieto säilyy eheämpänä ja se esiintyy vain yhdellä arvolla.
- Lisäksi tiedoille pystytään paremmin määrittelemään omistajuus, koska sama tietoyksikkö esiintyy vain yhdessä paikassa. Sen lisäksi pystytään parantamaan tietoturvaa, koska auktorisointi tarvitsee tehdä vain tiedon omistamassa järjestelmässä. Ts. tiettyjä tietoja ei pysty hakemaan eri paikoista, koska tieto on tallennettuna vain primäärijärjestelmään.
- **Yhteistyö ja tiedon vaihtaminen yhteistyökumppaneiden välillä**
- Tietomalli on stadardoitu ja tietoja pystytään koostamaan paremmin. Tällöin pystytään muodostamaan palveluja, joiden avulla tietoja voidaan lähettää ja vastaanottaa muista korkeakouluista (myös ulkomailta). Yhteistyökumppaneiden kanssa pystytään keskustelemaan samoilla käsitteillä ja tietojen vaihtaminen on mahdollista, jos yhteisestä skeemasta sovitaan.
- **Tiedon ajantasaisuus ja paikkansa pitävyys**
- Tietoa ei tarvitse synkronoida eri palveluiden välillä.
- **Tiedon monitorointi ja virhetilanteiden selvitys**
- Integraatioissa tarvitaan usein samantyyppisiä toimintoja, joita ovat esimerkiksi häiriötilanteesta toipuminen, häiriötilanteeseen liittyvien tietojen tallentaminen, häiriötilanteista tiedottaminen ylläpitäjälle, ajastustoiminnot ja jonotoiminnot). Tällä hetkellä näihin toiminnallisuuksiin liittyvää logiikkaa koodataan useaan paikkaan. Toteutukset

vaihtelevat ja ovat eri tasoisia. Palvelukerroksessa integraatiohin liittyvät yleiset toiminnallisuudet voidaan keskittää. Virhetilanteiden selvittäminen helpottuu ja nopeutuu, koska kaikki virheisiin liittyvät ilmoitukset löytyvät haluttaessa samasta paikasta. Kaikki viestit (sanomapohjaiset) pystytään tallentamaan, joten häiriötilanteissa pystytään selvittämään minkä viestin kohdalla häiriö on esiintynyt.

- **Raportoiminen**
- Raportteja pystytään luomaan palvelujen avulla ilman uusien järjestelmien luomista. Nämä palvelut pystyvät tarvittaessa koostamaan tietonsa palvelukerrokseen liitetyistä järjestelmistä.

Palveluväylän toiminta osana palvelupohjaista arkkitehtuuria

Palveluväylän rooli

- Viestin välitys palveluiden välillä
- Protokollien yhteensovittaminen
- Vanhojen järjestelmien käyttöönotto SOA-alustassa
- Viestityyppien muunnokset
- Tiedonkulun ja prosessien montorointi
- Palveluiden auktorisointi ja suojaus
- "Real-time data flow", Ei sovellu itsessään liiketoimintaprosessien kuvaamiseen/toteututtamiseen

Palveluväylään liittyviä käsitteitä ja teknologioita

Skeemat

- Ns. kanonisoitu malli, yhteinen skeema kuvaamaan palveluissa käytettäviä tietorakenteita ja näiden yhteyksiä toisiinsa.
- Versiointi ja siihen liittyvät käytännöt

Peruspalvelut

- Toisistaan riippumattomia palveluita, jotka eivät sisällä päällekäisiä toimintoja.

Registry

- Palveluiden kuvaukset löydettävissä yhdestä paikasta. LDAP, UDDI

Message-oriented middleware, MOM

- Alustoissa on otettu huomioon transaktioiden hallinta. MOM on matalan tason viestiliikenteeseen tehty alusta. JMS

Web Service mediation

- Viestimuunnokset
- Lokitus
- Tiedon suojaus

- Cache

Business process management (BPM)

- Liiketoimintaprosessien kuvaaminen standardilla tavalla
- "business process flow", vertaa data flow
- prosesit voivat olla pitkäkestoisia
- dehydraatiot/re-hydraatio, esim. Biztalk-palvelin voi suorittaa dehydraation jos vastausta ei tule 3 minuutin jälkeen.
- Standardi BPEL

Enterprise decision management

- Liiketoimintaprosesseja ohjaavia sääntöjä, joita voivat ylläpitää muut kuin tekniset asiantuntijat

Event proceccing and business activity monitoring (BAM)

- Tiedon seuraamiseen tarkoitettuja raportoinittyökaluja