Kohti palvelukeskeistä arkkitehtuuria

Palvelukerros
Proof Of Concept
selvitys Metropoliassa

RAPORTTI 30.9.2010
versio 1.1.

Koostanut:
Tapio Ekholm

Johdanto

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 rakentamaan sitä enemmän yhteyksiä, mitä enemmän kokonaisuudessa on järjestelmiä. Lisäksi tällä arkkitehtuurilla ei pystytä valvomaan riittävän hyvin tiedon eheyttä järjestelmien välillä.

Uutta palvelua kehitettäessä, siihen joudutaan 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 järjestelmätoimittaja 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 olemassa olevia.

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 saman tien 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.

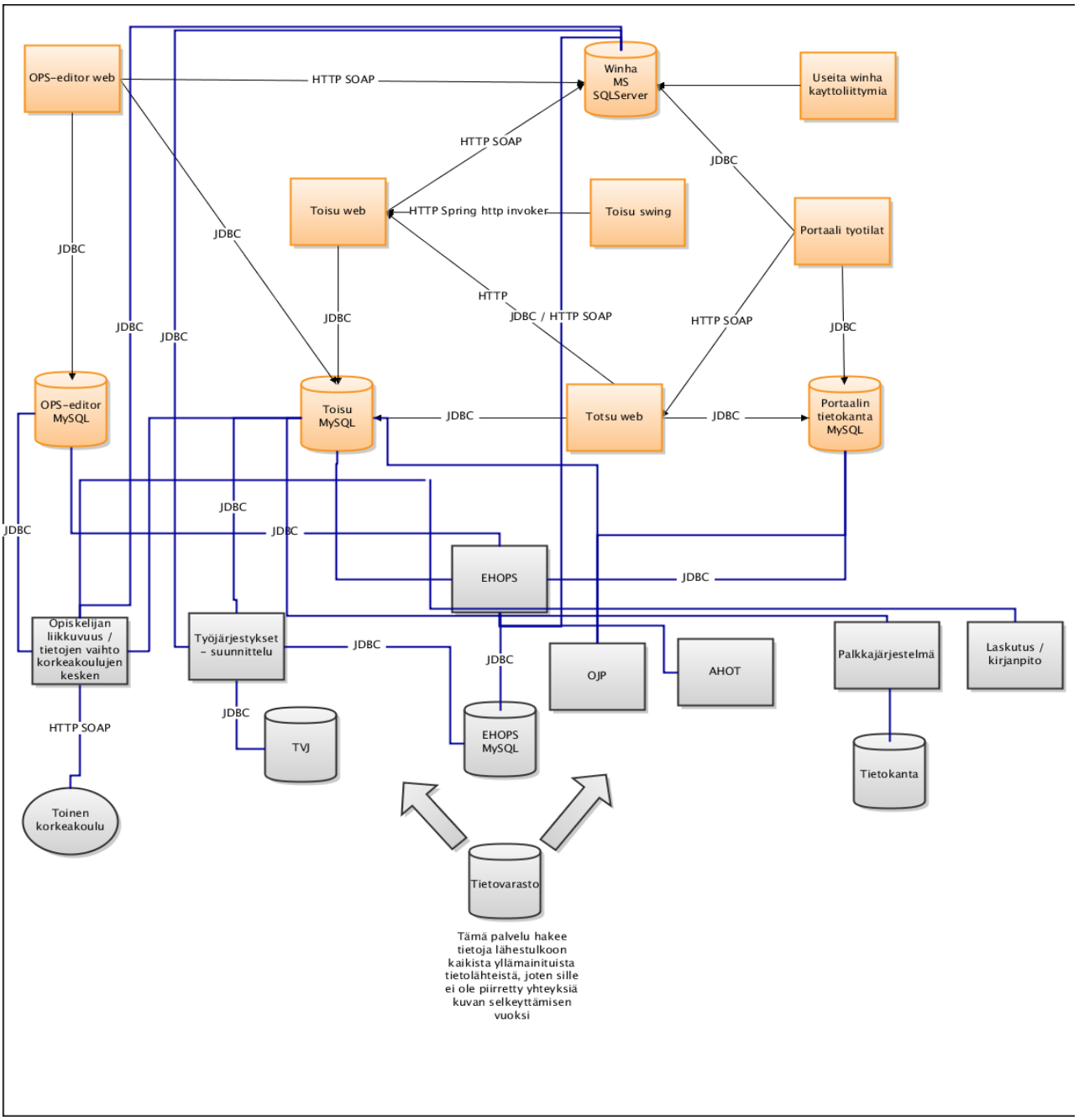
Raportointi

Nykyisellä arkkitehtuurilla joudutaan hakemaan raportointia varten tietoja monesta eri tietolähteestä ja tietoja joudutaan koostamaan. Lisäksi osa tiedoista on ristiriitaisia, koska sama tieto esiintyy monessa muodossa eri järjestelmissä, jolloin on raportoijan vastuulla selvittää kumpi tiedoista on oikea. Tämä aiheuttaa mm. kasvanutta työresurssien 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 yllä olevasta 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 riippuvuudet kasvavat entisestään. Lisäksi tulee huomioida, että uusien järjestelmien tuomat muutokset tulee ottaa huomioon myös niissä kaikissa olemassa olevissa järjestelmissä, joihin uusilla palveluilla tulisi olla yhteys.

Testauksen tavoitteet

Testauksen tehtävänä oli selvittää onko SOA-konsepti (Service Oriented Architecture) toimiva ja toteutettavissa oleva malli Metropolian tietojärjestelmien järjestelmäintegraatio-ongelmien ratkaisemiseksi.

Palvelukerros PoC testauksen (Proof Of Concept) konkreettisena kohteena oli selvittää voidaanko nykyiset vuosisuunnitteluun liittyvät palvelut uudistaa siten, että nykyiset "hämähäkki"-integraatiot poistetaan ja palvelut kytketään käyttämään palvelualustaa. Vaikka testaus kohdistui järjestelmien välisten integraatioiden toteutuksen, tulokset antavat viitteitä myös varsinaisen liiketoimintalogiikan toteuttamisesta palvelurajapintaa hyödyntämällä.

Testauskohteeksi valittiin Tuubin toteutuksen suunnitteluun liittyvän palvelun eli Totsu- palvelun uudistaminen, koska siihen liittyy nykyisessä muodossaan useita integraatioita eri järjestelmiin.

Testaukseen osallistuivat Tomi Blomqvist ja Simo Tiainen Metropoliasta sekä Eero Manninen toimittajan (Eduix) edustajana. Testauksessa tarvittavat skeemat ja niihin liittyvät impletoinnit teki Tuukka Puumala yhteistyössä Tapio Ekholmin kanssa.. Valtaosa raportin sisällöstä on syntynyt PoC prosessin aikana, josta olen sen kostanut.

Alustavassa selvitystyössä kartoitettiin mahdollisia palvelualusta tuotteita ja teknologioita, joista seuraava taulukko on yhteenvetona:

Tuote	Lisenssi	Tuetut standardit	Referenssien määrä	Teknologia	Muuta
OpenESB	Common Development and Distribution License (CDDL) Version 1.0 .	Java Business Integration (JBI, SOAP, WS-*	Useita kymmeniä referenssejä listattuna tuotteen sivuilla	.Java	Netbeans sisältää hyvät graafiset työkalut integrointiin ja palveluiden toteuttamiseen. 43 JBI-komponenttia. Seuraava versio tullaan näillä näkymin rakentamaan OSGI-teknologian päälle ( ). Project Fuji
JBoss ESB	ESB GNU General Public License, v. 2.0	SOAP	JBossESB-sivujen mukaan :"in use by a large insurance company for 3 years"	Java	Perustuu Jbossin JMS-teknologiaan (JBoss messaging)
BizTalk	Kaupallinen.	SOA		Net	
Apache Synapse	Apache license Version 2.0	SOAP, WS-*		Java	Axis 2-tuotteen päälle rakennettu
MuleESB	Common Public Attribution License Version 1.0 (CPAL)	SOAP	Yksi käytetyimmistä Open source -ESB tuotteista	Java	. Toimii yhdessä JBI-standardia tukevan tuotteen kanssa
ServiceMix	Apache license Version 2.0	Java Business Integration (JBI), hallintaan tuki JMX-teknologialle	Yksi käytetyimmistä Open source -ESB tuotteista. Käytössä apachen sovelluspalvelimessa Geronimossa.	Java	24 valmista JBI-omponenttia
WSO2 Enterprise Service Bus	Apache license Version 2.0	SOAP, WS-*		Java	Perustuu Apache Synapse -tuotteeseen

Testauksessa päätettiin nojautua JBI 1.0 standardia tukeviin tuotteisiin, koska niiden yhteensopivuus, odotettavissa oleva elinkaari ja soveltuvuus Metropolian teknologia-arkkitehtuuriin on todennäköisesti paras. Tällä perusteella testattaviksi palvelu/integraatio-tuotteiksi valikoituivat OpenESB ja ServiceMix.

Palvelualustalle asetettavia yleisiä vaatimuksia erilaisten skenaarioiden kautta

Termit

SOA, Service Oriented Architecture, palvelukeskeinen arkkitehtuuri
Palvelu, joukko yhteenliittyviä toimintoja, esim. opintosuunnitelman hakupalvelu.
Palvelualusta, tietojärjestelmä joka liittää palvelut yhteen.
Komponentti, palvelun toteutus.
Järjestelmä, palvelualusta ja siihen kiinteästi liittyvät muut tietojärjestelmät

Skenaariot eri roolien näkökulmista

Palvelun kehittäjä

Palvelun kehittäjä tarkoittaa sitä tahoa, joka vastaa uuden palvelun kehittämisestä, hankkimisesta tai käyttöönotosta tai joka osallistuu varsinaiseen kehitystyöhön.

Tilanne	Vaatimus	Ongelma jos vaatimus ei toteudu	PoC tulos
Yksittäinen palvelu on ollut käytössä 6 vuotta ilman että palvelua on jouduttu kertaakaan muuttamaan. Palvelu tarjoaa rajapinnan ja tätä vasten on tehty 8 kriittistä integraatiota, joiden tulee olla jatkuvasti toiminnassa. Palveluun on tarkoitus tehdä uusia ominaisuuksia sekä laajentaa ja muuttaa palvelun ulkoista rajapintaa.	Palvelun uusi versio on voitava asentaa tuotantoon vanhan tilalle siten, että vanhaa rajapintaa vasten tehdyt 8 integraatiota säilyvät ehjinä asennuksen jälkeen ilman muutoksia integraatioihin ja mahdollinen päivityskatko on mahdollisimman lyhyt.	Palveluiden kehittäminen vaikeutuu merkittävästi. Yhden palvelun päivitys voi tuoda mukanaan huomattavan määrän piilokustannuksia.	Yksittäisen palvelun rajapintaa ja palvelun olinpaikkaa on mahdollista muuttaa ilman että siitä koituu ongelmia siihen liitetyille integraatioille. Todettu tekemällä abstrakteja palvelukuvauksia, joihin muut palvelut voivat liittyä. Palvelun toteutus voidaan piilottaa täysin muilta palveluilta.
Liiketoiminta muuttuu ja tätä varten tarvitaan uusi palvelu tukemaan uutta tarvetta. Palvelulle löytyy valmis komponentti, mutta se on toteutettu eri tekniikalla/eri kielellä kuin muut alustaan liittyvät palvelut	Palvelu on voitava liittää palvelualustaan vaikka palvelun toteuttava komponentti on tehty eri tekniikalla kuin millä olemassa olevat komponentit on tehty.	Palvelua varten täytyy kehittää uusi komponentti vaikka tarjolla olisi valmis koeteltu komponentti. Pahimmassa tapauksessa komponentit voidaan ostaa vain tietyltä toimittajalta.	ESB-tuotteeseen on saatavilla useita komponentteja(binding component), joilla voidaan kytkeytyä erilaisiin rajapintoihin/protokolliin. Testattu mm. tiedosto, tcp/ip, http, soap, jdbc, ejb -komponentteja. Komponentin käyttöönotto vaatii integraatiossa hyvän dokumentaation/tuen työkaluille, koska sisältävät spesifejä konfigurointeja. JBI-komponentit ovat teoriassa liitettävissä toiseen JBI-standardia tukevaan tuotteeseen, mutta käytännössä kirjastoristiriidat estävät komponentin siirtämisen tai siirtämiseen liittyy riski rikkoa alustan toiminnallisuus.
Järjestelmään ollaan tekemässä uutta palvelua, joka toteutetaan koostamalla olemassa olevia	Järjestelmästa voidaan luoda testiympäristö sekä olemassa olevia palveluita vastaavat	Tuotantoon asennetaan virheellisesti toimivia komponentteja	Alustan palveluita on helppo korvata testaukseen tarkoitetuilla "stub"-palveluilla. Bpel-moduulissa

palveluita. Palvelun toiminta tulee testata kunnolla muita palveluita vasten ennen sen tuotantoon asennusta.	testipalvelut. Testien aikana tietovirtaa palveluiden välillä voidaan tarkastella.		on mukana lokitusominaisuus, jolla on mahdollista tehdä palvelimen lokiin tulostuksia muunnettavista sanomista. PoC-projektin aikana toteutettiin myös oma loki-palvelu, jolla voidaan tehdä keskitettyä lokitusta tietokantaan. Tietovirtaa on mahdollista seurata erillisillä tuotteilla, mutta PoC projektin aikana näitä ei ehditty tarpeeksi testaamaan.
Palvelun kehittäjä on tekemässä uutta hakupalvelua, mutta ei tiedä että sellainen on jo olemassa.	Palvelualusta voidaan listata siiheen liitetyt palvelut ja niiden toimintaperiaatteet.	Tehdään päällekäisiä palveluita, jolloin ylläpidon määrä ja kustannukset kasvavat.	Palvelimen hallintaliittymästä näkee kaikki asennetut komponentit sekä niiden palvelukuvaukset.
Palvelun kehittäjä tekee uutta palvelua ja huomaa että palvelussa voitaisiin hyödyntää olemassaolevia palveluita. Olemassaolevat palvelut eivät kuitenkaan tarjoa ulkoista rajapintaa.	Palvelualustaan on voitava tehdä palveluita siten, että muut palvelut voivat hyödyntää helposti palvelun tarjoamia rajapintoja. Vanhojen järjestelmien on pystyttävä kommunikoimaan palvelualustan kanssa tietokantayhteyden tai siirtotiedoston avulla.	Tehdään päällekäisiä palveluita, jolloin ylläpidon määrä ja kustannukset kasvavat.	Liittymäkomponenttien(Binding component) avulla on mahdollista julkaista palvelurajapinta tukeutuen siirtotiedostoihin tai tietokantatauluihin.
Palvelualustan kehitys ja tuki loppuu. Palvelualustaan on liittetty lukuisa määrä palveluita ja integraatioita.	Palvelualusta voidaan vaihtaa toiseen tuotteeseen kohtuullisella vaivalla ja ilman että palveluita joudutaan tekemään uudelleen.	Uusi palvelualusta ja integraatiot joudutaan toteuttamaan alusta alkaen, jolloin kustannukset ovat erittäin suuret.	Teoriassa onnistuu JBI-standardin avulla. Käytännössä nykyisillä tuotteilla tämä ei ole mahdollista tai siirtämiseen liittyy suuri riski. Ongelma liittyy kirjastoristiriitoihin. Seuraavan sukupolven tuotteissa tähän on tulossa parannus OSGI-teknologian myötä.
Palvelun kehittäjä on tekemässä integraatiota, joka johon liittyy 5 palvelua. Palveluiden rajapinnat eivät tue transaktioita. Palvelun kehittäjän tulee olla varma että häiriön sattuessa integraatio ei jää virheelliseen tilaan eikä tieto pääse vääristymään.	Palvelualustaan on voitava tehdä transaktionaalisia palveluita ja palvelualustan on tuettava hajautettuja transaktioita.	Monimutkaisia integraatioita ei voida tehdä tai ylläpito vaikeutuu merkittävästi. Integraation testaaminen on vaikeaa ja vauriolähtökohta kasvaa.	Kokeiltu transaktionaalisuutta OpenESB:ssä olevan bpel-komponentin avulla. Transaktiot suhteellisen uusi ominaisuus bpel-moottoreissa. Saatu transaktiot kuitenkin toimimaan, mutta tuotantokäyttöä ajatellen liikaa ongelmia. Palveluita suunniteltaessa huomioitava integraatioprosessit vs. liiketoimintaprosessit. Transaktiot soveltuvat lyhytkestoisiin integraatioprosesseihin ja prosessiin liittyvien rajapintojen täytyy myös tukea transaktioita. JOS palveluiden rajapinnat eivät tue transaktioita ei transaktionaalisuutta voi saavuttaa(korkeintaan kompensaatiota voidaan tehdä)
Palvelualusta ei toimi odotetulla tavalla.	Palvelualustaan liittyvän dokumentaation taso tulee olla hyvä sekä mahdollinen tuki helposti saatavilla. Palvelualustan toiminta voidaan tarkastaa alustan lähdekoodista ja tarvittaessa voidaan suorittaa pikakorjaus itse palvelualustan lähdekoodiin.	Palvelut joudutaan tekemään toisin kuin on suunniteltu. Palvelua ei voida toteuttaa. Pahimmassa tapauksessa koko palvelualusta joudutaan vaihtamaan.	Dokumentaatio puutteellinen. Integraatioissa ilmeneviä ongelmia voidaan jäljittää konfigurointitiedostosta ja lähdekoodista. Ongelman syyt ovat pääteltävissä ja usein korjattavissa. Versiossa 2.2 ilmennyt huomattavan paljon ongelmia netbeans-työkalussa. NetBeans-työkalun kehitys tulee myös loppumaan näillä näkymin SOAn osalta.
On tehty uusi palvelu, joka on tarkoitus asentaa tuotantoon.	Uuden palvelun asennus ei saa aiheuttaa käyttökatkoja muissa alustaan liitetyissä palveluissa.	Ylläpito vaikeutuu ja hidastuu. Järjestelmäpäivitykset vaikeutuvat.	Palveluita on mahdollista asentaa dynaamisesti jos asennuspaketti on eri. Abstraktien palvelukuvausten toimivuutta itsenäisesti asennettavien pakettien välillä ei ehditty tarpeeksi testaamaan.
Komponenttien kehittämiseen tarkoitettu kehitystyökalu toimii huonosti.	Komponentti voidaan kehittää/toteuttaa usealla eri kehitystyökalulla.	Uusien palveluiden kehittäminen on hidasta ja kallista.	OpenESB kehitykseen tarkoitettu työkalu (NetBeans) on liian buginen. Osa palveluista jouduttiin

	Komponentin rakenteen tulee olla niin yksinkertainen, että sitä voidaan tarkastella ilman erityistä kehitystyökalua. Komponentin onfiguraatiot ovat löydettävissä nopeasti.		tekemään aiemmalla versiolla. Paketointi ja konfiguraatioit ovat kuitenkin ymmärrettävissä, joten periaatteessa paketoinnin voi tehdä muillakin työkaluilla. Ajan puutteen vuoksi paketointia eri työkaluilla ei testattu.

Palvelualustan ylläpitäjä

Palvelualustan ylläpitäjällä tarkoitetaan tahoa, joka vastaa palvelualustan toiminnasta tai osallistuu sen ylläpitotyöhön.

Tilanne	Vaatimus	Ongelma jos vaatimus ei toteudu	PoC tulos
Palvelualustassa on tapahtunut vakava häiriötilanne.	Ylläpitäjän tulee saada tieto tapahtuneesta häiriöstä.	Palvelun käyttökatkopitenee	Alustaan on mahdollista tehdä lokituspalveluita. Quality of Service(QoS)-toiminnallisuutta testattiin, mutta viestin uudelleenlähetystä ei saatu toimimaan. Kokeiltu ajaa BPEL-moottoria persistoidussa tilassa ja havaittu että integraatiota yritetään uudelleen mikäli viestinkäsittelyssä tapahtuu virhe. Tämä voi kuitenkin myös generoida dublikaattiviestejä tietylle palvelulle, etenkin jos transaktioita ei ole käytössä.
Ylläpitäjä on saanut viesti vakavasta häiriötilanteesta.	Ylläpitäjä voi nähdä yhdestä paikasta järjestelmän kokonaistilan sekä järjestelmässä tapahtuneet häiriöt ja niiden aiheuttajat.	Ylläpito vaikeutuu ja aiheuttaa pitkiä käyttökatkoja.	OpenESB:n bpel-moottorissa prosessin viestit voidaan persistoida, joillin ne on löydettävissä tietokannasta. OpenEsb kirjoittaa lokia palvelimen lokiin. Monitorointiin liittyviä työkaluja ei ehditty testata tarpeeksi. Periaatteessa implementoitu lokituspalvelu voitaisiin hyödyntää keskitettynä virheviestien tallennuspalveluna.
Ylläpitäjä A on saanut ilmoituksen, että palvelu A ei toimi oikein. Ylläpitäjä B on saanut ilmoituksen, että palvelu B ei toimi oikein. Ilmoituksista ei kuitenkaan näe häiriön syytä. Palvelut A ja B on integroitu kolmanteen palveluun, joka aiheuttaa häiriön molemmissa järjestelmissä.	Ylläpitäjät näkevät helposti järjestelmien väliset riippuvuudet ja löytävät häiriönaiheuttajan nopeasti.	Ylläpitäjät tekevät turhaan työtä etsiessään häiriön syytä väärästä paikasta.	OpenESB:ssä Bpel-prosesseista on mahdollista generoida dokumentteja. OpenESB:n kehitystyökalut näyttävät riippuvuussuhteet.
Palveluiden käyttö kasvaa rajusti.	Palvelualustan suorituskykyä voidaan lisätä klusteroinnin avulla	. Koko palvelualusta joudutaan vaihtamaan tai osasta palveluita luopumaan.	Ei ehditty testaamaan.
Palvelualustaan on kytketty uusi palvelu, jossa oleva vika aiheuttaa liikaa kuormaa palvelualustalle säännöllisin väliajoin.	Palvelualustan hallintatyökaluilla voidaan tarkastella palvelun tilaa ja siihen liittyviä tietovirtoja. .	Ongelmien selvitys vaikeutuu. Kynnys asentaa uusia palveluita tai päivityksiä kasvaa	QoS-toiminnallisuuden avulla tietyn rajapinnan käyttäjäkuormaa voidaan rajata. Monitorointiin tarkoitettuja työkaluja ei ehditty riittävästi testata.
Palvelu joudutaan päivittämään. Palvelun rajapintoja käyttää kriittinen integraatio, joka lähettää palvelulle viestejä asynkronisesti. Integraation täytyy pystyä toimimaan huoltokatkon ajan.	Palvelun päivityskatkos ei saa aiheuttaa häiriötilannetta integraatioihin mikäli käytetään asynkronista tiedonsiirtoa.	Lisää ylläpitotyötä ja kynnys asentaa uusia palveluita tai päivityksiä kasvaa.	Bpel-moottorissa on mahdollisuus tallentaa prosessin tila käsittelyn ajaksi. Viestien korrelaatioita ei ehditty testata. Viestien korrelaatiolla ilmaistaan avaintieto, jolla persistoitu prosessi-instanssi voidaan ladata uudelleen

			muistiin(prosessit voivat olla usean kuukauden kestoisia liiketoimintaprosesseja)
Järjestelmään tehdään tietoturva-auditointi tai epäillään että järjestelmään on yritetty tehdä tietoturvahyökkäys.	Palvelualustaan liitetyt palvelut ja integraatiot voidaan listata ja tarkastaa niiden näkyvyys sisäverkon ulkopuolelle. Palvelualustassa tapahtuvaa tietovirtaa voidaan tarkastella. Ylläpitä saa kokonaiskäsityksen järjestelmässä käytössä olevista tietoturvaan liittyvistä teknologioista.	Riski hyökkäyksen onnistumisesta kasvaa, jos tietoturvan varmistaminen keskitetysti ei onnistu.	Asennettujen komponenttien rajapinnat ovat löydettävissä. OpenESB:ssä on mahdollista myös disabloida tiettyjä portteja ulospäin, tällöin palvelut ovat vain käytössä palveluväylän sisäisesti. Testatttu siten että tehty ejb-palvelu abstraktin palvelukuvauksen pohjalta ja disabloitu palvelun http-portti.
Palvelimelle on asennettu 3 palvelua, jotka on liitetty toisiinsa. Palvelimen osoite vaihtuu.	Palveluiden on pystyttävä olemaan yhteydessä toisiinsa vaikka palvelimen osoite vaihtuisi, ilman että palveluita joudutaan konfiguroimaan uudelleen.	Ylläpidon määrä kasvaa. Riski siihen, että jokin vanha palvelu hajoaa muutoksen yhteydessä on erittäin suuri.	Abstraktien palvelukuvausten avulla palvelut eivät tiedä toistensa fyysistä olinpaikkaa. Suurta palvelukokonaisuutta ei ilman ko. ominaisuutta pystytä järkevästi ylläpitämään.

Käytetty testiprosessi ja PoC vaiheet

Vaihe 1:

- skeeman luonti opintojaksototeutusta varten, päätetty toteuttaa MP-käsitemallin pohjalta
- OpenESB asennus
- nykyisen Toisun tietokannan ja HTTP-pyynnön liittäminen OpenESB:hen
- toisu-portletin konvertointi käyttämään OpenESB:n tarjoamaa rajapintaa
- testaus, aiheutetaan poikkeus kytkemällä Winha pois päältä ja tarkastetaan integraation tilanne

Vaihe 2:

- tehdään tulevaa Toisu-rajapintaa kuvaava transaktionaalinen palvelurajapinta
- tehdään Winha- rajapintaa kuvaava transaktionaalinen palvelurajapinta
- liitetään palvelut OpenESB:hen
- (toive, olisi hyvä toteuttaa myös loki-palvelu)
- testaus, aiheutetaan poikkeus kytkemällä Winha pois päältä ja tarkastetaan integraation tilanne

Vaihe 3:

- muutetaan Toisun rajapintaa
- testaus, tarkastetaan miten palvelua käyttävät toiminnot selviävät muutoksesta
- vaihdetaan OpenESB ServiceMix- tuotteeseen
- testaus, tarkastetaan miten helposti palvelut on siirrettävissä toiseen alustaan
- Raportoidaan PoC- projektin tulos
- Tulosten esittely

Aikataulu:

- 23.11.-27.11. ajalla järjestetään tekninen projektipalaveri
- Joulukuu, Metropolian käsitemallinpohjalta skeeman implementaatio (Tuukka, Tapio)
- Joulukuu, Metropolia asentaa kehityspalvelimen joulukuun loppuun mennessä (Mikko S./JKR)
- Tammikuu, projektipalaveri

- Tammikuu, toisella viikolla OpenESB asennus (Eero M., Tomi B., Simo T.)
- Tammikuu, kolmannella viikolla PoC-toteutuksen aloitus
- Helmikuu, Metropolian pojekti-ihmisten "rajapintakoulutus"
- Viikko 5, PoC testaus valmis
- projektipalaveri, yhteenveto
- Viikot 7 ja 8 raportin valmistelu
- Viikko 9, Tulosten esittely johtoryhmälle

Metropolian käsitemallinpohjalta skeeman implementaatio

PoC-projektia varten määriteltiin yleiset skeemat, jonka avulla viestien rakenne standardoitiin palveluiden välillä. Skeemat sijaitsevat education-planning projektissa.

Käytettävät skeemat

Projektissa käytettävä tietomalli noudattaa mahdollisimman pitkälle Raketti-hankkeessa tehtyjä tietomallimäärityksiä. Määrittelytyö on parhaillaan käynnissä, joten täysin valmiita malleja ei ole vielä saatavilla. Lisäksi malli on määritelty vasta käsitetasolle, joten toteutimme projektissa käytettävän XML-skeeman. Skeeman toteutuksessa noudatetaan suositusta JHS 170 Julkishallinnon XML-skeemat

Toteutetut skeemat ovat:

- Tietotyypit - yleisiä käsitteitä kuten henkilö, rooli, kieliteksti (lokalisoitava teksti) ja laajuus
- Opinto - Opintojen tietomalli, sisältää tällä hetkellä ainoastaan Opintojakso-elementin
- Opintototeutus - Toteutusten tietomalli, sisältää tällä hetkellä ainoastaan Opintojaksototeutus-elementin

PoC pilotointiympäristö

PoC varten pystytettiin virtuaalikone poc.metropolia.fi, johon asennettiin seuraavat tuotteet:

- OpenEsb v.2.2
- NetBeans
- Tehdyt konfiguraatiot:
- Tallennettu MySQL-ajuripaketti sovelluspalvelimen lib/ext kansioon ja määritelty tietokanta-asetukset sovelluspalvelimen admin-liittymässä.

Moduulit:

- ToisuEndpointImpl, toimii testirajapintana, joka tallentaa sanoman tiedot tietokantaan
- WinhaEndpointImpl, toimii transaktionaalisena rajapintana, joka aiheuttaa järjestelmäpoikkeuksen tietyn tyyppiselle viestille
- totsuService, bpel-palvelu transaktionaalisen bpel-prosessin testaamiseen. Tarjoaa soap-rajapinnan, jonka avulla viedään tiedot kahdelle eri palvelulle.
- RealizationStorageApp, paketoi totsuServicen asennuspaketiksi.
- RealizationService, bpel-palvelu, joka palauttaa toteutuksen tiedot tietyn skeeman mukaisena Toisun tietokannasta.
- RealizationSearchServiceCompApp, paketoi RealizationServicen asennuspaketiksi

Versionhallinta:

- trooper.metropolia.fi

PoC kuuluvien palveluiden toteutuksia:

Opiskelijarekisteripalvelu

Opiskelijarekisteripalvelulla tarkoitetaan Metropolian standardoidun toteutus-skeeman mukaisen tiedon siirtämistä konkreettiseen opiskelijahallintojärjestelmään. Oiskelijahallintojärjestelmänä Metropoliassa on Winha joka tarjoaa alunperin Toisua varten tehdyn Web Service -rajapinnan toteutusten siirtämiseksi Winhaan.

Kytkeytyminen Winhan rajapintaan

Winhan rajapinnan tekee erityisen ongelmalliseksi se, että se ei noudata täysin XML-standardia SOAP-viestin nimiavaruuksien määrittämisessä. Winha-rajapinta edellyttää, että nimiavaruus pitää olla määriteltynä SOAP- envelopen lisäksi myös itse lähetettävässä viestissä. Käytännössä kaikki standardia noudattavat työkalut optimoivat SOAP- envelopen generointia siten, että ylimääräiset nimiavaruuden määritykset poistetaan. Tästä syystä toimivan SOAP- envelopen generointi on tehtävä hyvin matalalla tasolla, jotta halutunlainen viesti voidaan muodostaa.

Rajapinnan epästandardina ominaisuutena voidaan nähdä myös se, että virhetilannetta kuvaava vastaussanoma palautetaan normaalina vastauksena, eikä SOAP- envelopen fault-elementin detail-osassa. Tämä ei ole varsinaisesti ongelma, mutta vastausviestinä käytetään sisältöä, mikä ajatuksellisesti kuuluisi eri paikkaan. Ongelmaksi tämä voisi muodostua siinä vaiheessa, kun ryhdytään toteuttamaan palveluja, mitkä myös palauttaa aidosti jotain sisältöä, koska tällöin joudutaan määrittelemään varsinainen vastaus ja virhesanoma samaan sanomaan.

Alla testatut liitymäkomponentit (binding component) ja niihin liittyvät ongelmat. Lopuksi esitetään toimiva ratkaisu kytkeytyä Winha-rajapintaan.

Liittymisestä Winha-rajapintaan.

SOAP-bindin component

Ongelmana SOAP- komponentissa on se, että lähtevä SOAP- envelope normalisoidaan ja nimiavaruus määritys puuttuu halutulta tasolta. Koska lähetettävää dokumenttia välitetään käytännössä olio-tietorakenteena, ei algoritmiin, mikä muodostaa xml-rakenteen merkkijonoksi, voi mitenkään päästä käsiksi.

HTTP-binding component

HTTP- liiittymäkomponentille (binding component) on mahdollista luoda SOAP- envelopelle halutunlainen merkkijonoesitys. Ongelmaksi muodostuu kuitenkin http-liittymäkomponentille välitettävän muuttujan tyyppi. Jos viestin lähettää liittymäkompontille merkkijonona, määräytyy http-pyynnön sisältö tekstityyppiseksi, mitä Winha- rajapinta palvelin ei ota vastaan väärän tyyppisenä viestinä. Jos muuttujan määrittää XML:nä, niin viestissä olevat "<" ja ">" -merkit muutetaan entiteettiviittauksiksi "<" ja ">". Koska http-pyynnön tyyppiä ei voi muuttaa riippumatta http-komponentille välitettävän muuttujan tyypistä, on oikeanlaisen http-pyynnön luonti mahdotonta.

TCP/IP- binding component

Komponentti edellytti yllättäen sisällökseen xml-muotoista viestiä, jolloin tällä liittymäkomponentilla http-pyynnön mukaisen viestin generoiminen osoittautui mahdottomaksi.

Toimiva tapa kytkeytyä Winhan rajapintaan

Kytkeytyminen Winhan rajapintaan ei onnistunut edellä kerrotun mukaisesti millään testatulla liittymäkomponentilla. Ainoaksi tavaksi keksimme toteuttaa kytkeytymisen EJB- komponentilla, mikä julkaistiin palveluksi openESB:n käyttöön. Eli huonosti käyttäytyvä rajapinta piilotettiin ("wräpättiin") toimivan rajapinnan alle. Itse kytkeytyminen toteutettiin Apache Commons:in http-kirjaston avulla käyttäen suoraan http-protokollaa. Apachen http- rajapinat tarjosi mahdollisuuden määrätä http pyynnön sisällön tyyppi täysin sisällöstä riippumatta, mikä mahdollisti http-liittymäkomponentista poiketen toimivan ratkaisun toteuttamisen.

Tämä wräpätty rajapinnan toiminta muutettiin alkuperäisestä rajapinnasta siten, että se palauttaa virhesanoman SOAP- envelopen fault-elementin detail-osassa.

Opiskelijarekisteripalvelun logiikan toteutus

Opiskelijarekisterin logiikkaa (ts. Metropolian standardin skeeman muuntaminen Winhan- skeemaan) lähettiin toteuttamaan kahdella eri openESB:stä löytyvällä Moottorilla (service enginellä): XSLT service engine ja BPEL-service engine.

XSLT-service engine component

XSLT-service engine:llä voidaan tehdä palvelu siten, että voidaan määrittää XSLT- muunnos lähtevän viestin ja paluuviestin väliin. Näin tehden XSLT- service engine toimii siltana kohdejärjestelmän ja kutsujan välissä, missä viestiliikenne käy mennen tullen läpi XSLT- muunnoksen. Johtuen Winhan ongelmista tästä ratkaisusta luovuttiin, koska service enginen toimintaa on vaikea lokittaa. Tästä syystä siirryimme käyttämään BPEL-service engineä.

BPEL-service engine component

Opiskelijarekisteripalveluun liittyvä logiikka on toteutettu BPEL- moottorilla siten, että varsinainen skeemamuunnos tehdään XSLT- muunnoksella. Muunnos tehdään XSLT:llä sen takia, että BPEL:n tarjoamaa mäppäystyökaluja ei voitu käyttää wsdl:n puutteellisten skeema-määritysten takia. Ongelmana on se, että palvelun wsdl- kuvaus ei määrittele vastausviestin sisältämiä elementtejä mitenkään, joten niihin ei voi mitenkään BPEL- moottorissa viitata. XSLT- muunnoksessa voidaan viitata elementtiin, vaikka sitä ei skeemassa olisikaan määritelty.

Virheen kaappausta BPEL- enginessä ei saatu käytettävissä olevan ajan puitteissa toteutettua. Tavoitteena oli, että virhesanoman sisällä palautettu virhesanoma olisi saatu ohjattua esim. tiedostoon. ongelmaksi jäi, että virheen palauttava muuttuja ei syystä tai toisesta alustunut catch- lohkossa, vaikka catch.- lohko aktivoitui asianmukaisesti.

XSLT- muunnos ei huomio lähdeaineistosta kuin opinnon koodin ja toimipisteen. Nämä kaksi kenttää riittävät kuitenkin osoittamaan rajapinnan periaatteellisen toimivuuden. Muokkaamalla XSLT:tä pidemmälle, voidaan kyseisellä toteutuksella saada aikaiseksi täysin toimiva siirto, mikäli keskitetyn skeeman ja Winha- rajapinnan skeeman välillä ei ilmene ristiriitaisuuksia (Tätä en ole järjestelmällisesti varmistanut).

Asennusohje

Opiskelijarekisteripalvelu on riippuvainen kolmesta alla luetellusta PoC-repositoryn projektista. Asennus tapahtuu ajamalla deploy komposiittiprojektille: studentAdministrationAdministrationServerComposition.

winhaWrapper3: EJB- pavulla toteutettu Winha- rajapinnan "wräppäävä" palvelu.

studentAdministrationService: BPEL moottorilla toteutettu varsinainen opiskelijarekisteripalvelu

studentAdministrationAdministrationServerComposition: Kahden edellisen projektin sisältävä komposiitti-projekti, mikä asennetaan palvelimelle.

Demo

Syötettävä aineisto voidaan kirjoittaa tiedostoon: studentAdministrationService\inputFilesDataa\inputRealization.xml ja ajaa bat- tiedostolla studentAdministrationService\inputFiles\siirraInputRealization.bat Winhaan. (Bat-tiedosto tekee olennaisesti edellä mainitun tiedoston kopioinnin inputFiles- kansioon, mitä studentAdministrationService pollaa.

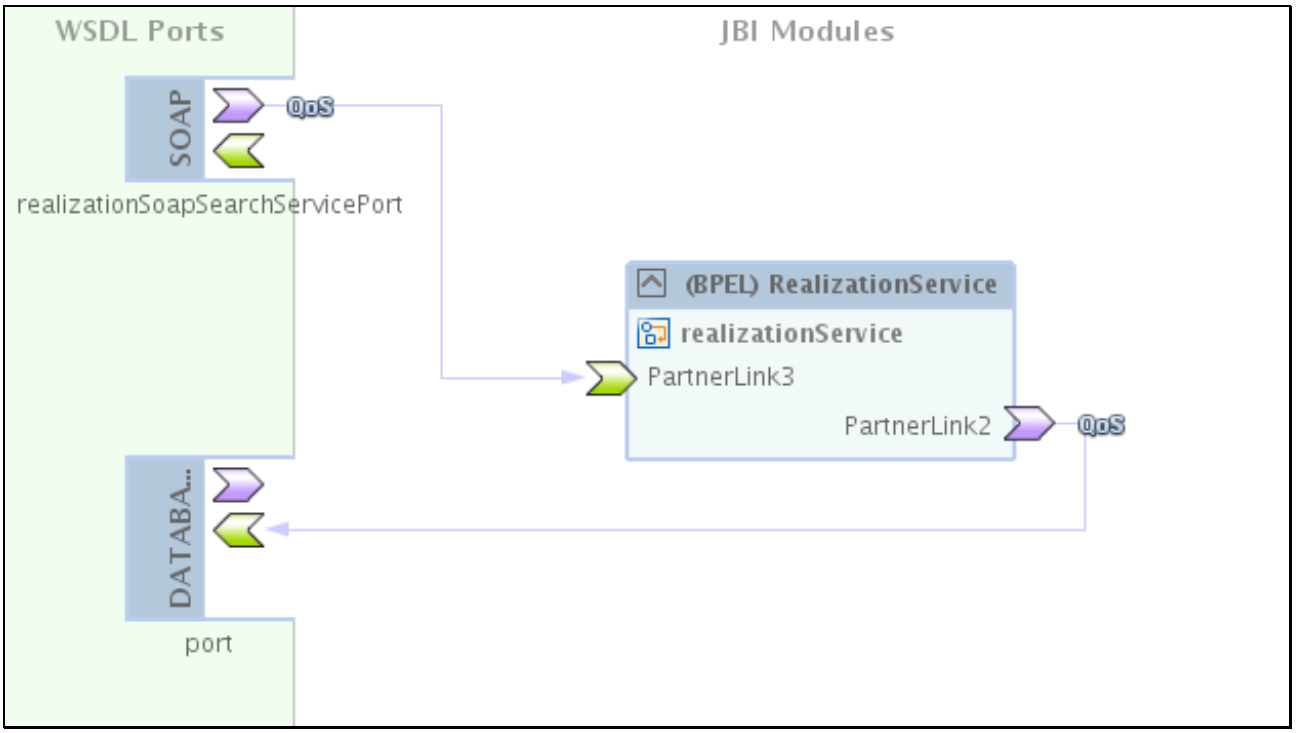
Palvelu Toisua varten

Moduulin tarkoituksena on testata miten Metropolian nykyisten järjestelmien tietokannasta voidaan hakea tietoa ja julkaista ne uudelleen käyttäen yleisiä skeemoja.

Moduulin komponentit

- BPEL-prosessi, joka on yhteydessä Toisun kantaan tietokantakomponentin(database binding component)avulla, muuntaa tietokannan tiedot yleisen skeeman mukaiseksi ja julkaisee SOAP-rajapinnan
- totsu-portlet, joka hakee toteutuksen tiedot SOAP- rajapinnalta ja näyttää ne käyttöliittymässä.

Tietokannan kytkeminen OpenESB:hen onnistui ilman ongelmia. Netbeans sisältää hyvät ja toimivat työkalut palvelun käyttöönottoon.



Transaktionaalisen palvelun testaus

Moduulin tarkoituksen testata OpenESB:n BPEL- moottorin kyky käsitellä transaktoita.

Moduulin komponentit:

- Transaktionaalinen EJB-tekniikalla toteutettu rajapinta, joka tallentaa viestin tietokantaan(kuvaa toisu-rajapintaa)
- Transaktionaalinen EJB-tekniikalla toteutettu rajapinta(kuvaa Winha-rajapintaa), jos viestissä tietty merkkijono, aiheuttaa Runtime poikkeuksen, jolloin transaktion tulee peruuntua.
- BPEL-prosessi, joka ottaa vastaan SOAP-viestin, aloittaa transaktion ja kutsuu järjestyksessä ensin toisu-rajapintaa ja tämän jälkeen winha-rajapintaa.
- Muutettu totsu-portlet lähettämään viesti OpenEsb:n julkaisemaan SOAP-palveluun

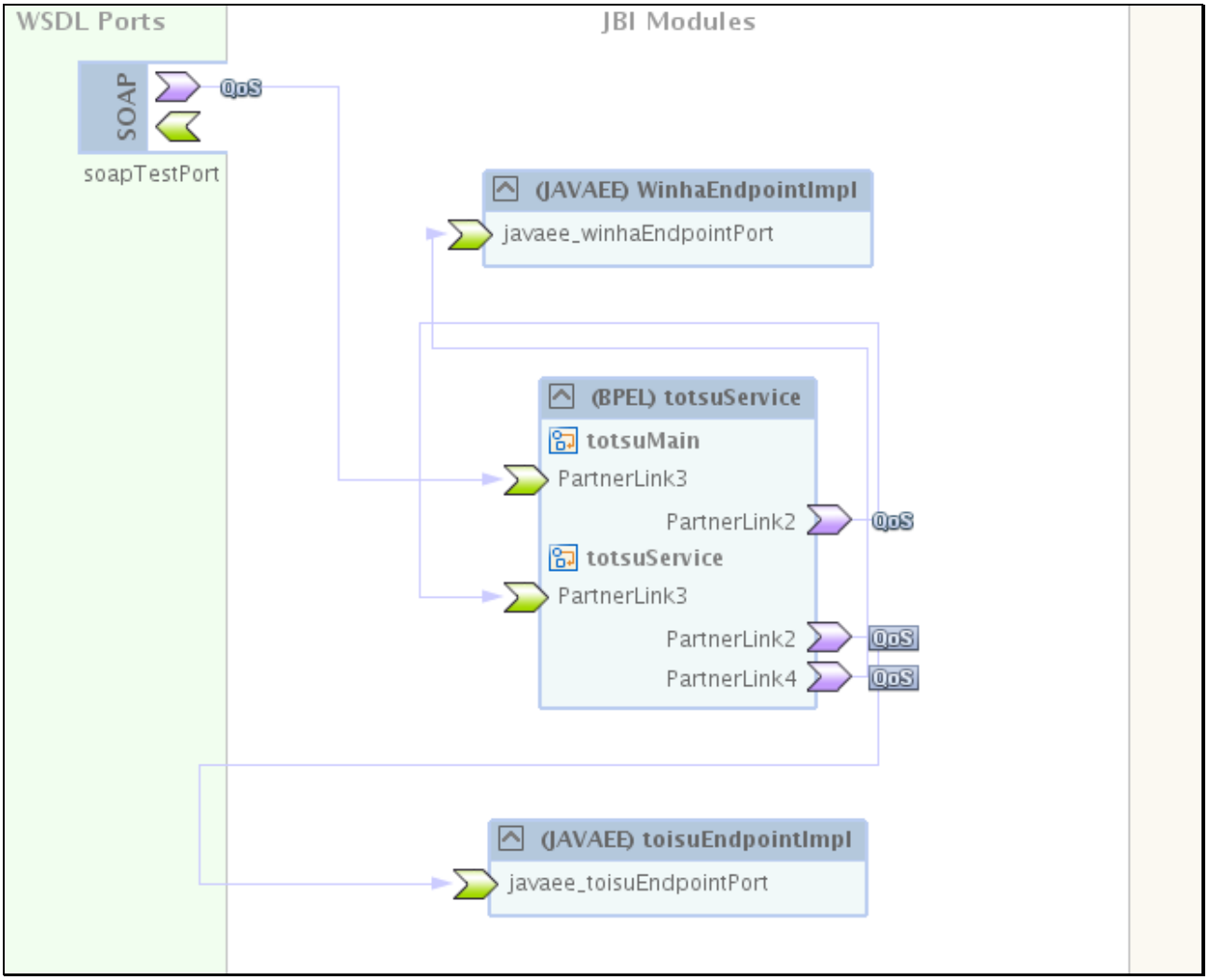
Testin kulku

Lähetetään totsu-portletista viesti SOAP-palveluun
Saadaan vastaus ja tarkastetaan tietokannan tilanne
Todetaan että tieto on tallentunut Toisu-pavelun tietokantaan

Lisätään lähetettävään viestiin merkkijono "throw_exception", jolloin Winha-palvelun pitäisi heittää Runtime-poikkeus
Tarkastetaan sovelluspalvelimen loki, jossa kerrotaan transaktion epäonnistuneen
Tarkastetaan että tietoa ei löydy Toisu-palvelun tietokannasta

Testin tulos

Transaktiotesti onnistui, mutta vaikka uudessa 2.2 versiossa voi konfiguroida tiedon(XAtransaction attribute), jonka mukaan transaktioita luodaan, täytyy sisään tulevan viestin olla joko liitettynä transaktioon tai BPEL -prosessi täytyy käynnistää toisen BPEL -prosessin sisältä. Lisäksi BPEL -moottori tulee olla konfiguroitu persistoituun tilaan sekä palvelurajapintojen täytyy olla transaktionaalisia. Transaktion peruuntumisen jälkeen lokiin tuli myös virheviestejä liittyen BPEL -moottorin tietokantayhteyteen. Kyseisessä versiossa liikaa ongelmia transaktioiden toimintaan saamiseksi, jotta sitä voisi suositella tuotantokäyttöön.



Yhteenveto

PoC- projektin aikana tehtiin integraatioita OpenESB tuotteella. Projektin aikana käytettiin useita protokollasidonnaisia komponentteja, joilla testattiin tuotteen integroitavuutta. Lähtökohta oli hakea vastauksia aiemmin esitettyihin skenaarioihin ja integraatio-ongelmiin. Projektin aikataulua haittasi huonosti toteutettu Winha- rajapinta, jonka kytkeminen OpenESB:hen osoittautui odotettua suuremmaksi haasteeksi. Lisäksi uuden version työkalut havaittiin liian bugisiksi tuotantokäyttöä ajatellen ja projektin aikana toteutui yksi skenaarioissa mainituista uhkakuvista, tuotteen tulevaisuus on kyseenalainen OpenEsb tuotteen tekijän SUNin myyminen Oraclelle. Toimittajasidonnaisuus huomioitiin alusta alkaen ja testattaviksi tuotteiksi valittiin JBI- standardia noudattavat tuotteet. Komponenttien siirrettävyyteen liittyy kuitenkin nykyisillä versioilla riskejä, kirjastoristiriidat, joten käytännössä standardi ei tällä hetkellä takaa toimivuutta.

Lukuun ottamatta OpenESB- tuotteeseen liittyviä ongelmia, SOA- konsepti todettiin toimivaksi ja on toteutettavissa. Todettiin että ESB- tuotteen ja yleisen tietorakenteita kuvaavan skeeman avulla päästään eroon järjestelmien välisistä riippuvuuksista.

Projektien aikana kertyneitä huomioita

Palveluiden määrittelyissä on haasteena pitää palvelut sopivan kokoisina. Jos palvelut ovat liian pieniä (yksinkertaisia metodikutsuja), koosteisista palveluista saattaa tulla liian hitaita. Jos palvelut liian suuria, uudelleenkäytettävyys laskee ja palvelun ylläpito vaikeutuu, pienen korjauksen takia suuri joukko toimintoja täytyy asentaa uudelleen. Jotkin toiminnot voivat olla liian hitaita tuotavaksi ainoastaan palveluväylän kautta loppukäyttäjille. Huonosti toimiva ja määritetty rajapinta vaatii käytännössä uuden palvelun, jolla vain kääritään huonosti toimivan palvelun rajapinta toimivaksi. Integraatiotuotteiden(OpenESb, ServiceMix, BizTalk) komponentteja ei helposti saa toimimaan huonosti tehdyn rajapinnan(esim. Winha-rajapinta) kanssa. Palveluiden rajapintojen tulisi käyttää häiriötilanteissa mahdollisimman standardia tapaa kuvata tilanne, esimerkiksi SOAP-sanoman fault-elementti.

ESB-tuotteella on mahdollista saada viesti varmasti välitettyä, Käytännössä tapahtuu niin että ESB pitää viestiä tietokannassa lähetyksen ajan. Tämä ei kuitenkaan takaa tiedon virheettömyyttä, viesti voi tulla häiriötilanteesta riippuen palvelulle useampaan kertaan ja sotkea tiedonkulun. Ainoastaan hajautetut transaktiot takaavat viestin lähetyksen vain yhteen kertaan.

Julkaistut rajapinnat tulee erottaa palvelun sisäisistä rajapinnoista. Huomioitava että tietorakenteet ovat osa rajapintaa ja tätä varten sovittava yleisesti käytössä olevat skeemat tietorakenteille. Julkaistujen rajapintojen versiointiin ja ylläpitoon täytyy olla selkeät käytännöt.

Lyhyt yhteenveto tuloksista

- SOA- konsepti on toimiva ja toteutettavissa.
- palvelurajapinnan hyödyntäminen kannattaa aloittaa järjestelmien välisten integraatioiden toteuttamisesta
- pitkäkestoisiin prosesseihin liittyvä liiketoimintalogiikan toteuttaminen vaatii lisäselvitystä
- lopullisen toteutusalustan valinnassa kannattaa vielä odottaa, mutta tuotteen tulee kuitenkin tukea JBI- standardia
- vaihtoehtoisena ratkaisumallina voi olla myös SCA (Service Component Architecture) specifikaatiota tukevat tuotteet, joita PoC:ssa ei testattu

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

Ehdotus konkreettisista jatkotoimenpiteistä

- Logican toimittamien rajapintojen uusiminen standardin mukaisiksi
- Rajapinta käyttäjäkoulun käynnistäminen Metropoliassa projektipäälliköille, projektiin osallistuville sekä päättäjille
- Skeematyötä ja siihen liittyvää impletointia tulee jatkaa ja laajentaa (myös osallistumalla kansallisiin hankkeisiin)