Sovellusten liittäminen BPEL-prosessiin

Markku Hirvonen

14.5 2008

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

Tiivistelmä

Liiketoiminnan mallintaminen ja mallinnetun prosessin suorittaminen BPEL-kielen avulla on eräs tapa hallita palveluorientoituneen arkkitehtuurin eli SOA:n mukaisia palveluista rakentuvia sovelluksia. Sovelluksien, myös BPEL-sovellusten, yhdistäminen voidaan suorittaa käyttämällä konfiguraatiolähtöistä tapaa, jossa kytketään sovelluksia, lähinnä liiketoiminta- tai asiakasyhteyksienhoitosovelluksia, muunnospalvelut, reitityksen ja sovittimet eli adapterit sisältävään ”yrityspalveluväylään”. Ohjelmointilähtöisesti sovelluksia voidaan muodostaa esimerkiksi luomalla ensin web-palvelu, liittää palvelu BPEL-prosessiin partnerilinkin avulla ja lopuksi tarvittaessa integroida BPEL-sovellus yrityspalveluväylän avulla osaksi räätälöityä yrityskohtaista ratkaisua.

Perinnesovelluksia, joihin sisältyy monesti käyttäjien aikaisempaa osaamista, on joskus järkevää säilyttää osana yrityksen ohjelmistoratkaisua esimerkiksi uusien sovellusten korkeiden oppimiskynnysten vuoksi. Tällainen perinnesovelluksen liittäminen voi tapahtua esimerkiksi C/C++-kieltä käyttävällä *gSOAP*-työkalulla.

Tutkielmassa tarkastellaan Oraclen BPEL-palvelimen mahdollistamaa työlistasovellusta sekä perinnesovelluksen liittämistä BPEL-prosessiin web-palveluna sekä tämän käyttämistä työlistasovelluksessa.

ACM-luokat (ACM Computing Classification System, 1998 version): D.2.6, H.3.5, H.4.1

Avainsanat: BPEL, web-palvelut, integroitu kehitysympäristö, perinnesovellukset, työnkulku

Esipuhe

Tämä pro gradu -tutkielma sai alkusysäyksen keväällä 2007 tekemästäni *systemoinnin menetelmien* -kurssin harjoitustyöstä, jonka aiheena oli käyttäjän tehtävän liittäminen BPEL-prosessiin, ja on jatkoa aiheesta syksyllä 2007 tekemääni kandidaatintutkielmaan. Kiitän Raimo Raskia kärsivällisyydestä ja syventymisestä aiheeseen tutkielmien ohjauksessa kuluneen vuoden aikana. Lisäksi esitän lämpimän kiitoksen sisarelleni Marjatta Karkkulaiselle kannustuksesta matkalle tietojenkäsittelytieteen opintoihin.

Sisältö

Kuvat

1 Johdanto

BPEL on kehittyvien Web-standardien, kuten XML-skeema, SOAP ja WSDL, päälle rakennettu kieli liiketoimintaprosessien orkestroimiseen ja suorittamiseen. BPEL on osa ns. *palvelusuuntautunutta arkkitehtuuria (Service Oriented Architecture, SOA)*, jonka toteutus voi perustua web-palveluihin.

Liiketoimintaprosessien suorituskieli web-palveluille (*Business Process Execution Language for Web Services* eli *BPEL4WS*) julkaistiin vuoden 2002 heinäkuussa ensimmäisen kerran BPEL4WS 1.0 spesifikaationa IBM:n, Microsoftin ja BEA:n yhteistyön tuloksena. Määritelmässä ehdotettiin webpalvelujen orkestrointikieltä, joka oli edellisten, IBM:n WSFL:n (Web Service Flow Language) ja Microsoftin XLANG:n pohjalle määritelty. Toukokuussa 2003 versio 1.1 julkaistiin ja tällöin mukaan olivat tulleet SAP ja Siebel Systems (Barreto et al., 2007).

BPEL-kielen käyttö on käytännössä objektien, *aktiviteeteiksi* nimitettyjen prosessisymbolien, ”drag and drop” siirtämistä BPEL-prosessin ohjelmavirtaan. Sellaisena se on ”laajaa” *(programming in the large)* ohjelmointia, ts. rakenteet ovat abstrakteja kokonaisuuksia. Tyypillinen rakenne on esim. assign-aktiviteetti, joka voi olla XPath-standardilla määritelty XML-dokumenttiin kohdistuva kopiointitoimenpide. Aktiviteetti voi olla myös varsin laaja kokonaisuus, palvelu, kuten käyttäjän aktiviteetti *(user task activity)*, tietokanta- tai tiedostoaktiviteetti. Laajan aktiviteetin vieminen ohjelmavirtaan käynnistää tavallisesti asetusvelhon, johon saatetaan syöttää lukuisia asetusparametrejä. Palveluaktiviteetit voidaan myös määritellä ns. partnerilinkkeinä, jolloin palvelun tai BPEL-prosessin sijainti voi fyysisti olla jossakin muualla kuin BPEL-palvelinkoneessa.

Liiketoimintaprosessien luonne on tavallisesti lähes mekaanisia, automaattisia toimenpiteitä vaativa. Joskus nämä prosessit, vaikka ne parametriensa puolesta voisikin toteuttaa automaattisina, vaativat kuitenkin käyttäjän puuttumista tai hyväksymistä. Tällainen prosessi on esimerkiksi lainahakemuksen käsittely.

Oasiksen (http://www.oasis-open.org) *WS-BPEL Technical Overview for Developers and Architects - Part 1* (Ryan, 2007) esittää integraation olevan avainongelma liikeyrityksille erityisesti eri yrityssovellusten keskinäisessä integraatiossa ja liikepartnerien integraatiossa *(business-to-business)*. Muita integraatiota eteenpäin vieviä seikkoja ovat tarve koostaa palveluja omalle toimialueelle ja web-palvelujen siirtyminen kohti

palveluorientoitunutta laskentaa, jolle ominaista on: sovelluksia tarkastellaan palveluina, väljästi kytketty dynaaminen vuorovaikutus, heterogeeniset alustat ja yhdelläkään osallisella ei ole täydellistä hallintaa (Ryan, 2007).

Syynä BPEL:n kaltaisen kielen tarpeeseen on, että WSDL:n kuvaamilla webpalveluilla on tilaton *(stateless)* vuorovaikutusmalli eli viestejä vaihdetaan käyttäen joko synkronisia tai korreloimattomia ei-synkronisia invokaatioita. Useimmat tosimaailman liiketoimintaprosessit kuitenkin tarvitsevat vankemman viestien vuorovaikutusmallin eli sellaisen, jossa viestejä vaihdetaan kaksisuuntaisella ”peer-to-peer” -tiedonvaihdolla, joka saattaa kestää pitkänkin ajan (minuutteja - kuukausia, jne.). BPEL:n avulla on mahdollista kuvata tilallista *(stateful)*, pitkään kestävää vuorovaikutusta viesteissä (Ryan, 2007).

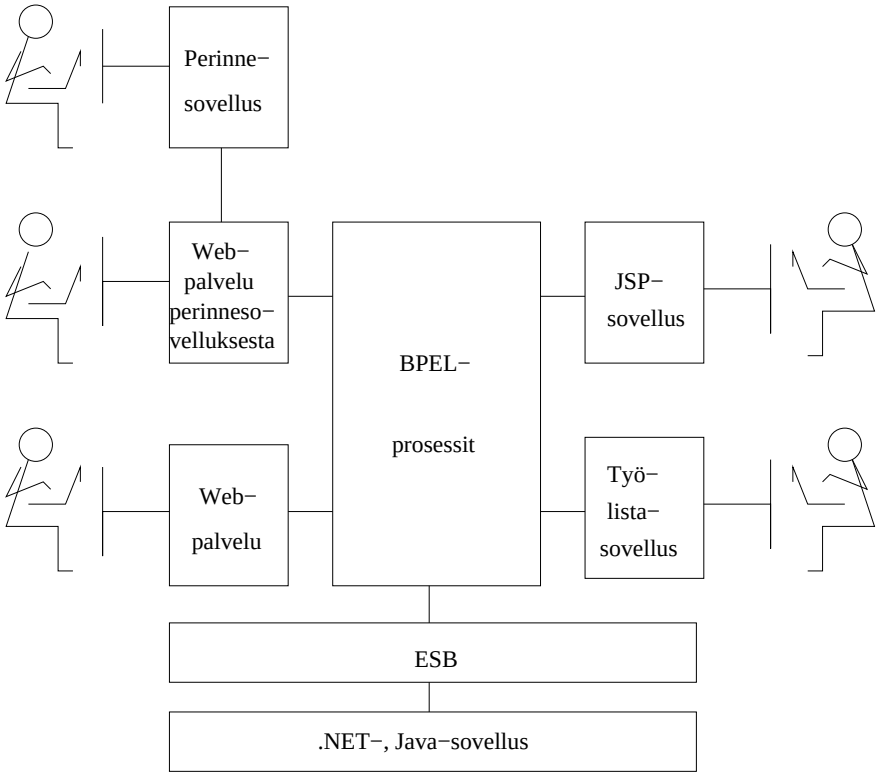
Ryanin mukaan BPEL:n etuja on, että BPEL on teollisuusstandardi, sopii luonnostaan webpalvelupinoon (kuvattu kokonaan XML:llä, käyttää ja laajentaa WSDL-standardia ja käyttää XML-Schema 1.0 datamallia), siirrettävyys alustojen ja ohjelmistotuottajien välillä ja toiminnallisuus vuorovaikutteisten prosessien välillä.

BPEL kategorisoidaan yrityssovellusten integraatioon (*Enterprise Application Integration, EAI*), johon kuuluu myös yrityspalveluväylä (*Enterprise Service Bus eli ESB*). Esimerkiksi Oraclen SOA Suite -paketti sisältää väylästä implementaation, joka toimii välittäjänä (*mediator*) BPEL:stä, asiakkaiden suhdetoiminta- (CRM) ja yritysten resurssien hallintaohjelmistoista (ERP) .NET:iin, Javaan ja perinnesovelluksiin. SOA Suiten osana olevalla ESB:llä on viesti-, reititys- ja muunnospalvelut, jotka mahdollistavat integroinnin.

ESB:n ohella perinnesovelluksia (*legacy application*) voidaan integroida ohjelmoijaläheisimmin työkaluin. Perinnesovelluksien, jotka ovat vielä käyttökelpoisia, joita on taloudellisesta syistä järkevää käyttää edelleen ja joihin liittyy paljon käyttäjien osaamista, integroiminen BPEL-pohjaisiin tai muihin web-palveluihin voidaan toteuttaa luomalla olennaisista sovellusten osista liittymät web-palveluina esim. sellaisia työkaluja kuten *gSOAP* ja Apachen *Axis2c* avulla.

Ohjelmoijan näkökulma BPEL-työkaluihin, erityisesti *JDeveloperia* käytettäessä, on joko luoda automatisoituja tai manuaalisia BPEL-prosesseja (Hirvonen, 2007a; Hirvonen, 2007b) . Manuaaliset BPEL-prosessit ovat yleensä käyttäjän työlista-rajapintoja soveltavia JSP- ja servletti-sivuja. BPEL-standardia moitittiin aluksi käyttäjäaktivitee-

tin puuttumisesta ja useat alan ohjelmistojen tuottajat (Active Endpoints, Adobe, BEA, IBM, Oracle ja SAP) esittelivätkin sitten *BPEL4Peoplen*, jossa tavoitteena on luoda laajennus, jossa ihmisaktiviteetti on käytettävissä (Agrawal et al., 2007; Silver, 2006).



Kuva 1: Sovellukset ja BPEL-prosessi

Tässä tutkielmassa käsitellään kuvan 1 esittämän idean mukaisesti Oraclen sovelluspalvelinalustalle rakennettua BPEL-palvelinympäristöä ja syvennytään ohjelmoijan kannalta kiinnostaviin aiheisiin kuten työkulkupalveluihin, käyttäjän ja BPEL-prosessin vuorovaikutukseen ja palvelutyyppiseen käyttäjäliityntään (työlista-API). Keskeisinä alueina ovat perinnesovellusten liittäminen BPEL-prosessiin, BPEL-prosessimanageri ja sen ylläpitämät oheispalvelut kuten työnkulku, ilmoituspalvelu sekä työlistasovellus. Tutkielmassa oletetaan BPEL-kieli lukijalle tunnetuksi.

Luvussa 2 tuodaan esille SOA-arkkitehtuuriin liittyviä käsitteitä ja luvussa 3 tarkastellaan Oraclen BPEL-prosessimanageria, BPEL-palvelinta, BPEL-konsolia ja tavalli-

simpia vuorovaikutusmuotoja BPEL-prosessin ja sovellusten välillä. Luvuissa 4 ja 5 syvennytään työnkulkupalveluihin ja luvussa 6 integroidaan SOAP:n avulla C-kielellä toteutettu palvelu BPEL-prosessiin. Luku 7 on yhteenveto käsitellystä aiheesta.

2 Palvelusuuntautuneesta arkkitehtuurista

IBM:n SOA-raportissa todetaan huomionarvoiseksi topologisten kaavojen historiallinen kehitys palvelujen yhdistämiseksi alkaen point-to-point verkosta, siirtyen sitten keskuslähtöiseen palvelujen yhteyksien välitykseen (*hub-and-spoke*) ja lopuksi väylämalliin (*bus*). Väylämalli viittaa siihen, ettei mitään erityistä yhdistettävyyden keskuspistettä ole olemassa, vaan esittelee ajatuksen, että on olemassa rakenne, joka läpäisee koko hajautetun verkon (High et al., 2005).

Edellytyksenä SOA:n soveltamiseen informaatioteknologiassa onkin *löyhän kytkennän (loose coupling)* periaate. Periaate tarkoittaa väliaikaisten, teknologisten ja organisaationaalisten rajoitteiden välttämistä tai ainakin kapseloimista informaatiojärjestelmän suunnittelussa. Sama periaate pätee palvelun määritelmään. Karkeana yleistyksenä *palvelu* on toistuva tehtävä liiketoimintaprosessissa. Jos siis voidaan tunnistaa liiketoimintaprosessit ja niissä suoritettavat tehtävät, voidaan väittää, että tehtävät ovat palveluja ja liiketoimintaprosessi on *palvelujen koostuma (composition of services)* (High et al., 2005).

Liiketoimintaprosessien integraatio koostuu esimerkiksi kuvan 2 komponenteista. Web-sovellukset edustavat käyttäjän (loppukäyttäjän tai liiketoimintaprosessien tuottajien) liittymää muihin komponentteihin. Työnkulku organisoi liiketoimintaprosessien hallintaa, tietokannat säilövät tietoa kuin myös web-sisältöä ja perinnesovelluksia voidaan liittää osaksi palveluja tai tuoda palveluja niihin esimerkiksi ESB-väylän kautta. Liittymä voidaan luoda myös liiketoiminnan transaktioformaattia EDI:ä *(Electronic Data Interface)* käyttäviin järjestelmiin.

Kuva 2: Integraatio Ryanin (2007) mukaan

2.1 EAI

Lyhenne EAI (*Enterprise Application Integration*) tarkoittaa rajoittamatonta tiedon ja liiketoimintaprosessien jakamista organisaation verkkosovellusten tai tietolähteiden kautta (Webopedia, 2004a). Varhaiset ohjelmistot sellaisilla alueilla kuten inventaariohallinta, ihmisresurssit, myynnin automaatio ja tietokannan hoito oli suunniteltu toimimaan itsenäisesti ilman vuorovaikutusta järjestelmien välillä. Ohjelmistot olivat erityisesti rakennettuja sen ajan teknologialla ja olivat lisäksi suljettuja kaupallisia järjestelmiä. Yritysten kasvaessa ja tunnistaessa tarpeen siirtää ja jakaa informaatiota ja sovelluksia järjestelmien välillä yritykset sijoittavat EAI:hin tarkoituksena virtaviivaistaa prosesseja pitäen kaikki yrityksen elementit yhdistettynä.

EAI:sta on neljä eri kategoriaa:

- *Tietokannan linkittäminen* eli tietokannat jakavat informaatiota ja kahdentavat tietoa tarvittaessa.
- *Sovellusten linkittäminen* eli yritys jakaa liiketoimintaprosesseja ja tietoa kahden tai useamman sovelluksen välillä.
- *Tiedon varastoiminen (data warehousing)* eli tietoa irrotetaan moninaisista tietolähteistä tietokantoihin ja analyysiin.
- *Tavanomaiset virtuaaliset järjestelmät* eli EAI:n painopiste: kaikki yrityksen tietotekniikka-aspektit sidotaan yhteen siten, että ne voivat esiintyä yhtenäisenä sovelluksena.

2.2 WS-pino

Ryanin (2007) esityksessä *webpalvelujen pinoksi* kutsuttu kuvaus jakaantuu neljään kerrokseen jotka koostuvat seuraavasti:

1. Liiketoimintaprosessit
    - *WS-BPEL*. WS- eli web-palvelu on ohjelmistojärjestelmä joka mahdollistaa tietokoneiden välisen vuorovaikutuksen http- tai internet-pohjaisen tietoverkon yli käyttäen www-rajapintoja. Web-palvelun osat ovat *tarjoaja (provider)*, *käyttäjä (requester)* ja *palveluhakemisto (broker)*. Kommunikointi tapahtuu XML-pohjaisten protokollien avulla. BPEL on web-palvelujen orkesterointistandardi, joka perustuu XML-pohjaiseen kielioppiin orkesterointilogiikan kuvaamiseksi web-palvelujen välillä liiketoimintaprosessissa
2. Kuvaus
    - *WSDL* on webpalvelujen kuvauskieli esim. SOAP-protokollan mukaisten rajapintojen muodostamiseen.
    - *Menettelytavat (policy)* tarjoavat mekanismit esitellä web-palvelujen kyvyt ja vaatimukset.
    - *UDDI eli Universal Description, Discovery and Integration*, on nimi ryhmälle web-pohjaisia rekistereitä, jotka paljastavat informaatiota liiketoiminnasta *(business)* tai muusta entiteetistä ja sen teknisistä rajapinnoista

(API:sta). Näitä rekistereitä ylläpidetään usean operaattorisivuston *(Operator Site)* avulla, ja niitä voi käyttää jokainen joka haluaa saattaa tietoa saataville yhdestä tai useammasta liiketoiminnasta tai entiteetistä, kuin myös jokainen joka haluaa saada tietoa tästä tiedosta. Näista peruspalveluista ei peritä maksua (Bellwood, 2002).

- *Tarkastus (inspection)* tarjoaa XML-formaatin palvelujen tarkastamisen avuksi.

3. Palvelun laatu *(Quality Of Service)*

   - *Turvallisuus (security)* eli osapuolten luotettava tunnistaminen esim. sertifikaattien avulla.
   - *Luotettava viestintä (reliable messaging)* eli viestien koskemattomuus ja SSL-kerroksen käyttö.
   - *Transaktiot (transactions specifications)* määrittelevät tuen hajautetuille tapahtumille web-palvelujen kanssa.
   - *Koordinaatio (coordination)* määrittelee web-palveluiden koordinointikehyksen.

4. Kuljetus ja enkoodaus

   - *SOAP (Simple Object Access Protocol)*: XML-pohjainen kommunikaatioprotokolla.
   - *XML (Extensible Markup Language)*: muunnokset sovelluksista XML:ään ja päinvastoin.
   - *Muut protokollat ja palvelut*: esim. Java-protokollat.

2.3 ESB

Kirjassa *Enterprise Service Bus* todetaan (Chappel, 2004):

> IT-ammattilaiset olivat olleet pettyneitä joidenkin edellisten teknologiatrendien kuten CORBA:n ja EAI:n suhteen. CORBA:lla oli oikea ajatus SOA:n näkökulmasta, mutta osoittautui, että se oli liian monimutkainen soveltamiseen ja ylläpitoon johtuen sen tukeutumisesta tiukasti kytkettyihin (*tighly coupled*) liitäntöihin sovellusten ja palvelujen välillä..

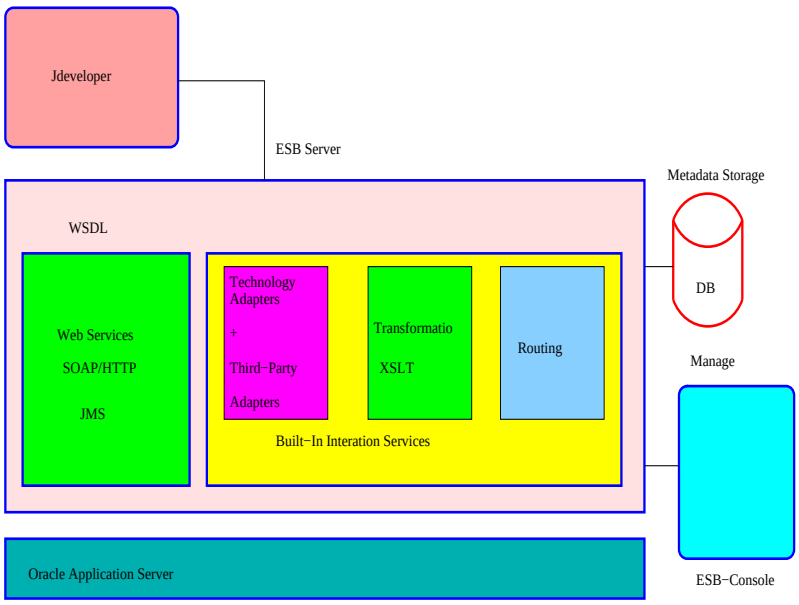
> EAI myös kärsi jyrkistä oppimiskäyristä ja kalleista muureista yksittäisten projektien aloituksiin. Mitä tarvittiin oli yksinkertainen lähestymistapa SOA:aan, arkkitehtuurilla joka voitaisiin sovittaa sopimaan mihin tahansa integraatioon yrityksissä, pieneen tai suureen. Lisäksi tarvittiin kestävä arkkitehtuuri, joka oli kykenevä kestämään evoluutiota protokollissa, liityntäteknologioissa ja jopa prosessien mallintamistrendeissä. ESB konsepti luotiin käsittelemään näitä tarpeita.

ESB on lyhenne sanoista *Enterprise Service Bus* tarkoittaen myös viestien välittäjää (*message broker* tai *mediator*) (Webopedia, 2004b). ESB on avoimiin standardeihin perustuva hajautetuttu synkroninen tai asynkroninen viestejä välittävä väliohjelmisto (*middleware*) ja tarjoaa turvallisen keskinäisen operoinnin yrityssovellusten ja webpalvelujen rajapintojen välillä XML:n kautta sekä stardardiin perustuvan dokumenttien reitittämisen. Käytännössä siis datatiedostoja siirretään edeltä sovittujen ohjenuorien avulla, jotka ovat yhteisiä kaikille osapuolille sen varmistamiseksi että tieto säilyy koskemattomana, kun sitä reititetään. Kaksi tavallista hajautettujen järjestelmien arkkitehtuuria ESB:ssä ovat J2EE ja .NET. ESB on vanhemman EAI:n laajennus ja on lisännyt tähän:

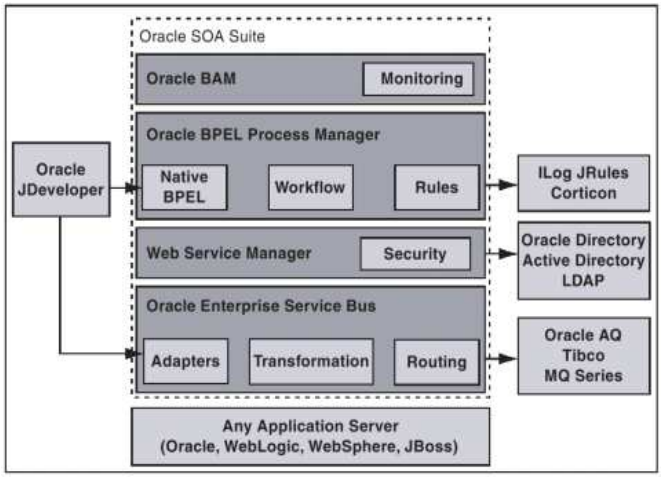
- *XML-muunnokset*
- *siirrettävyyden* erilaisissa tietokonejärjestelmissä
- *kuorman tasauksen (load balancing)/klusteroinnin* eli käsittelyn hajauttamisen useille laittelle
- *viestifunktioden siirrettävyyden toiselle palvelimelle* järjestelmän pettäessä (failover).

ESB:n avainpiirre on tarjota perusta (*underpinning*) tukea hajautettujen löyhästi liitettyjen liiketoimintayksikköjen ja liiketoimintakumppaneiden automatisoituja hankintaketjuja (*supply chains*). ESB:hen kuuluukin käsite *laajennetusta yrityksestä (extended enterprise)*, joka kuvaa organisaatiota ja sen liikekumppaneita, jotka on erotettu liiketoiminta- ja fyysisillä rajoilla. Laajennetussa yrityksessä jopa sovellukset, jotka ovat yksittäisen korporaation hallinnassa, voivat olla erillään maantieteellisen hajanaisuuden, korporaatioiden palomuurien ja osastojenvälisten turvallisuuskäytäntöjen kautta (Chappel, 2004).

Wikipedian (2008a) mukaan ESB on *ohjelmistoarkkitehtuurikonstruktio* kun taas IBM määrittelee sen *arkkitehturaaliseksi malliksi (pattern)* (High et al., 2005). ESB:n ensisijaisena etuna lähestymistavassa on vähentää päästä-päähän (*point-to-point*) yhteyksiä, joita tarvitaan sovellusten kommunikaatioon. Ihannetapauksessa ESB:n pitäisi korvata kaikki suora kontakti sovellusten välillä väylän kautta tapahtuvaksi viestinvälityksekseksi. Sovelluksen ollessa perinnesovellus ESB:n on muutettava viestit sovellukselle sopivaan muotoon. Ohjelmiston osaa, joka hoitaa tämän, kutsutaan *adapteriksi*.



Kuva 3: Oraclen ESB-toteutus (Dinesh et. al., 2006)

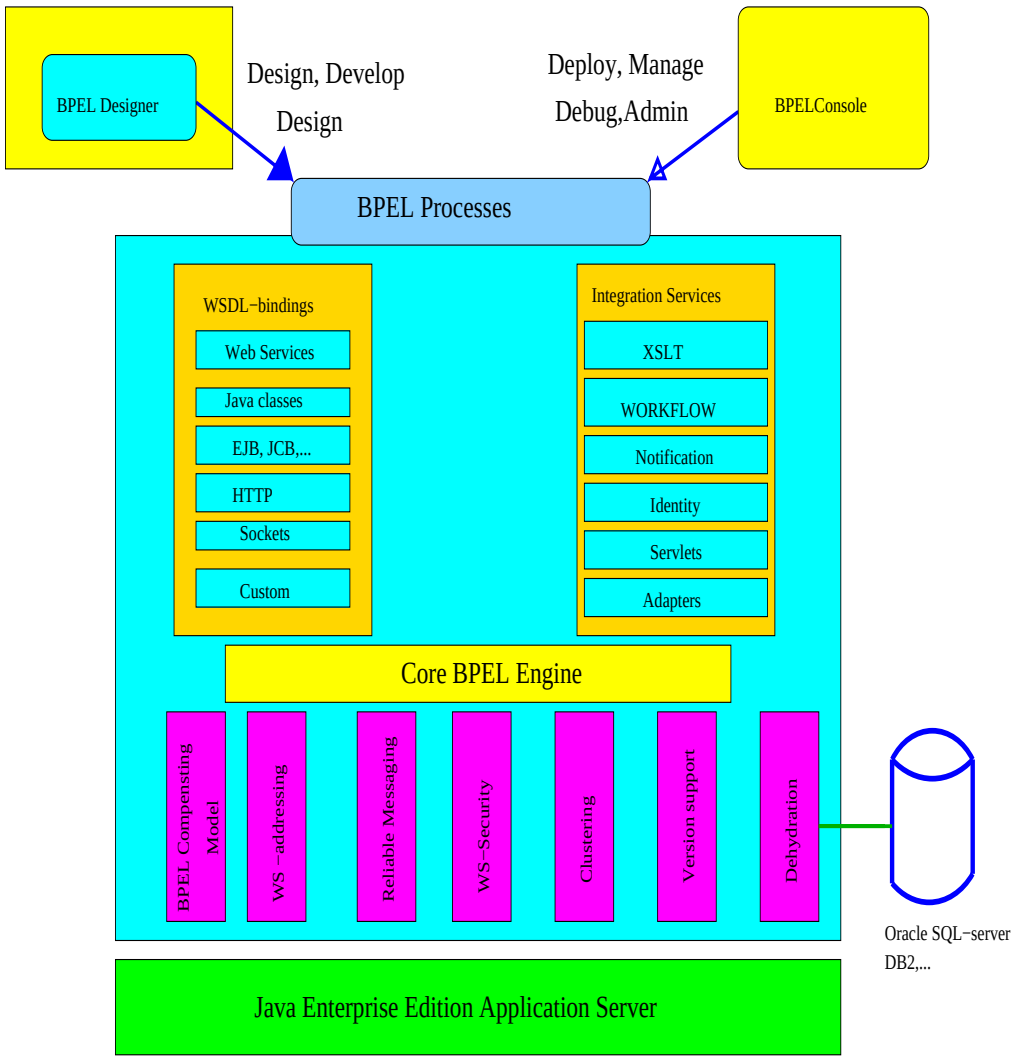


Kuva 4: ESB ja BPEL-prosessimanageri Oraclen SOA-kokoonpanossa (Dinesh et. al., 2006)

Artikkelissaan *ESB-oriented architecture: The wrong approach to adopting SOA* Woolf (2007) nimeää ESB:n ongelmaksi sen, ettei se itsessään tuota lisäarvoa, vaan se on keino saavuttaa tavoite, eli liiketoiminta ei tuota ennenkuin järjestelmät on yhdistetty ja toimivat yhdessä. Artikkelin päätelmässä Woolf arvioi SOA:n olevan paremman vaihtoehdon verrattuna ESB-orientoituneeseen arkkitehtuuriin ja ehdottaa ESB:n rakentamista osaksi SOA:a.

Kuvat 3 ja 4 havainnollistavat Oraclen ESB-palveluja ja SOA-kokoonpanoa. Osa Oraclen BPEL-palvelimesta tuttua toiminnallisuutta on mukana SOA-suiten ESB-osassa. Protokollasidonnat HTTP/SOAP, JMS, JCA, WSIF ja Java löytyvät ESB:stä, mutta ESB-palvelin ei tue RMI:tä. Samalla tavalla kuin BPEL-konsolia käytetään kontrolloimaan BPEL-prosesseja tapahtuu palvelujen hoito ja hallinta ESB-palvelimeen rekisteröidyille palveluille webpohjaisessa ESB-konsolissa (Dinesh et al., 2006).

3 Prosessimanageri



Kuva 5: BPEL-prossessimanagerin arkkitehtuuri (Juric et. al., 2004)

Oraclen BPEL Process Manager eli prosessimangeri (ks. kuva 5), aiemmalta nimeltään Collaxa BPEL Server, on ajonaikainen ympäristö BPEL-prosesseille. Manageri tukee 1.1-versiota BPEL-spesifikaatiosta ja siinä on mukana työkaluja prosessien käyttöönottoon, tarkkailuun ja ylläpitoon. Prosessimanageri on toteutettu Javalla toimien J2EE-yhteensopivalla *sovelluspalvelimella (application server)*. Oletussovelluspalvelin on Oracle Application Server OC4J (Oracle Containers for Java). Lisäksi on mahdollista käyttää avoimen lähdekoodin JBoss- ja BEA:n Weblogic -sovelluspalvelimia. Manuaalisella asennuksella lisäksi Sunin ja IBM:n sovelluspalvelimia voidaan käyttää (Juric et al., 2004).

Itse prosessimanageri koostuu neljästä osasta: BPEL-palvelin, BPEL-konsoli, BPEL Designer ja tietokanta.

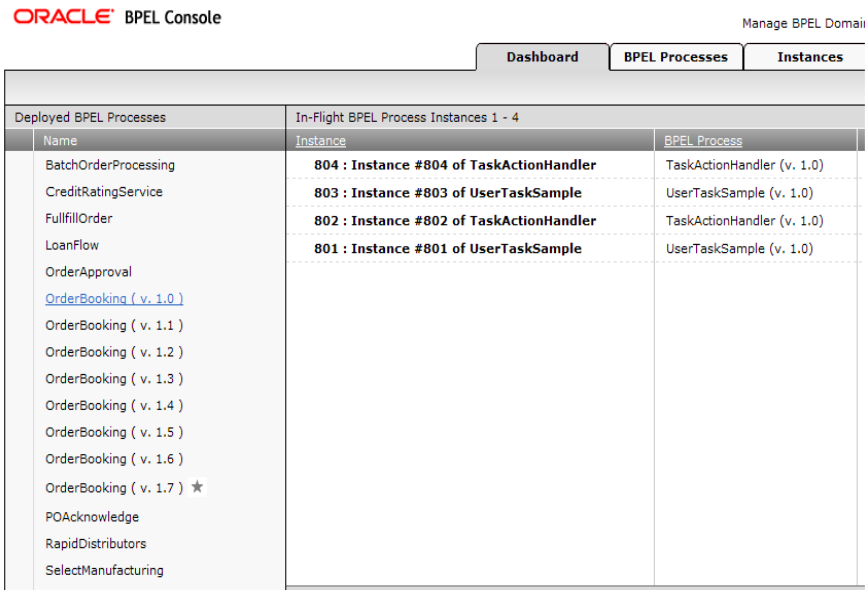
3.1 BPEL-konsoli

BPEL-konsolin (ks. kuva 6) avulla voidaan ajaa, seurata ja hallita BPEL-prosesseja, jotka on luotu ja otettu käyttöön (deployed) joko *JDeveloper BPEL Designer* -välineellä tai *Eclipse BPEL Designer* -välineellä. Myös BPEL-domaineja voidaan hallita konsolista käsin (Bradshaw et al., 2005b).

3.2 BPEL-palvelin

Palvelin koostuu seuraavista osista (Juric et al., 2004) :

- *BPEL-moottori*
- *WSDL-sidonnat*
- *integraatiopalvelut*



Kuva 6: Oraclen BPEL-konsoli (Bradshaw et. al., 2005b)

3.2.1 BPEL-moottori

BPEL-moottori on ajonaikainen ympäristö, jossa BPEL-prosesseja *otetaan käyttöön (deploy)* ja suoritetaan. Muita ominaisuuksia moottorissa ovat tuki esim. ”web palvelujen osoitteille” *(WS-addressing), versionhallinta* käyttöön otettaessa prosesseja, pitkäkestoisten prosessien *dehydraatio* eli prosessien tilan säilöntä (tietokantaan), jossa ne eivät kuluta ajoympäristön voimavaroja, ja klusterointi (Juric et al., 2004).

3.2.2 WSDL-sidonnat

WSDL-sidontakehys vastaa palvelimen käynnistämien BPEL-prosessien tiedonvaihdosta. Se sisältää asiakkaat, jotka haluavat pääsyn muihin BPEL-prosesseihin ja BPEL-prosessit, jotka haluavat päästä kosketukseen muihin webpalveluihin eli partnerilinkkeihin. Oraclen palvelin voi SOAP-protokollan lisäksi käyttää tiedonvaihtoon *EJB (Enterprise Java Beans), RMI (Remote Method Invocation), JMS (Java Message Service), JCA (Java Connector Architecture)* ja sähköposti- sekä http-protokollia (HTTP GET ja POST). Integraatio on toteutettu Apachen *WSIF:n (Web Services Invocation Framework)* (http://ws.apache.org/wsif/) kautta (Juric et al., 2004).

3.2.3 Integraatiopalvelut

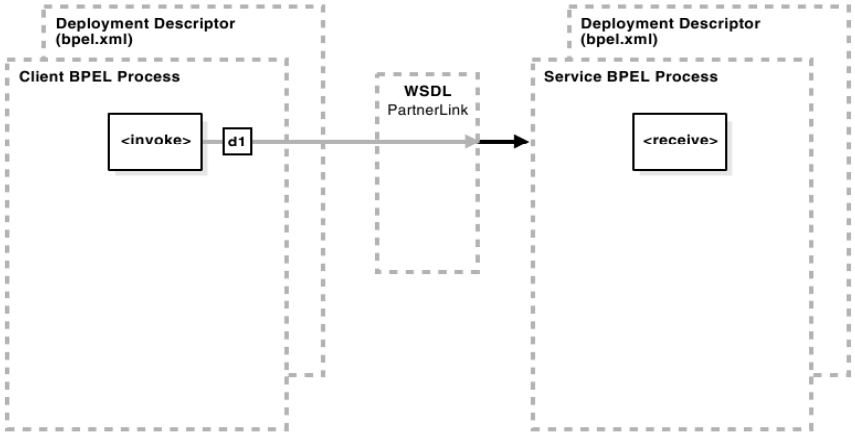
Integraatiopalvelut sisältävät *XPath*-tuen lisäksi muunnospalveluina *XSLT-, XQuery-* ja *XSQL*-muunnokset. BPEL-palvelimessa on Java-integraatio, jossa on kaksi mahdollisuutta, joko käyttää upotettua Java-koodia BPEL-prosesseissa tai käyttää WSIF:ä. Toiminnallisuus saavutetaan joukolla rajapintoja. Edelleen integraatiopalveluihin luetaan työnkulku-, ilmoitus- ja identiteettipalvelut. BPEL-prosessien integroimiseksi tiedostoihin, ftp-palveluihin ja tietokantatauluihin, JMS:ään tai Oracle-sovelluksiin, BPEL-palvelin sisältää tuen ns. *teknologia-adaptereihin*. Adaptereilla päästään käsiksi edellisiin WSDL-liityntöjen kautta. Adapterit perustuvat *JCA 1.5*-resurssiadaptereihin ja WSIF:iin (Juric et al., 2004).

3.3 Tavallisimmat vuorovaikutusmuodot BPEL-prosessin ja sovelluksien välillä

BPEL-prosessit voivat toimia sekä asiakkaina että palveluina. Seuraavassa on lueteltu tavallisimpia vuorovaikutusmuotoja ja parhaimpia käytäntöjä niiden soveltamiseen (Bradshaw et al., 2005c).

3.3.1 Yksisuuntainen viesti

Yksisuuntaisessa viestissä *(One-Way Message)* asiakas lähettää viestin palvelulle eikä palvelimen tarvitse vastata. Viestimuodossa tarvitaan partnerilinkki, invoke-aktiviteetti ja receive-aktiviteetti. Kuva 7 havainnollistaa yksisuuntaista viestiä.



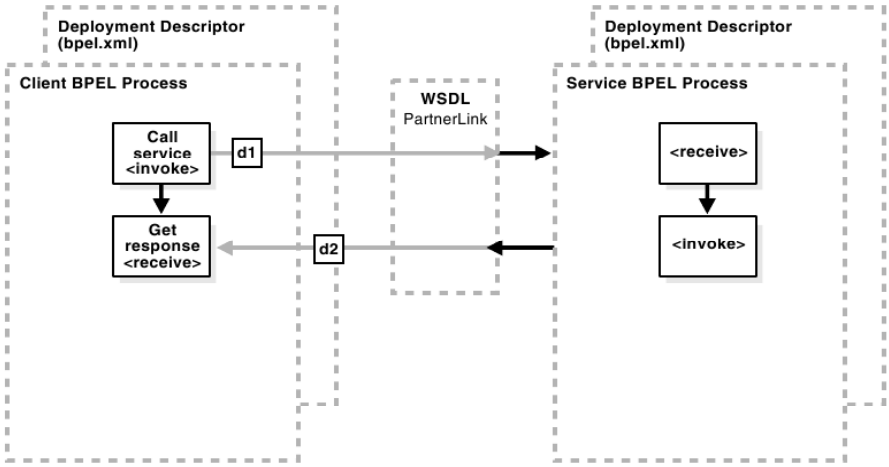
Kuva 7: Yksisuuntainen viesti (Bradshaw et. al., 2005b)

3.3.2 Synkroninen vuorovaikutus

Synkronisessa vuorovaikutuksessa asiakas lähettää viestin palvelulle ja saa välittömän vastauksen. BPEL-prosessi voi olla vuorovaikutuksen molemmissa päissä ja ne on koodattava eri tavalla riippuen niiden roolista.

3.3.3 Asynkroninen vuorovaikutus

Asynkronisessa vuorovaikutuksessa asiakas lähettää pyynnön palvelulle ja odottaa kunnes palvelin vastaa. Molemmat päät tarvitsevat invoke- ja receive-aktiviteetin, joskin eri järjestyksessä riippuen aloittajasta. Kuva 8 havainnollistaa asynkronista vuorovaikutusta.



Kuva 8: Asynkroninen vuorovaikutus (Bradshaw et. al., 2005b)

3.3.4 Asynkroninen vuorovaikutus ajastimella

Aikavalvontaa käyttävässä asynkronisessa vuorovaikutuksessa asiakas lähettää pyynnön palvelimelle ja odottaa, kunnes palvelu vastaa tai kunnes tietty aikaraja saavutetaan. Kun BPEL-prosessi on asiakkaana, tarvitsee se invoke-aktiviteetin lähettääkseen pyynnön ja pick-aktiviteetin kahdella haarautumisella, jossa vaihtoehtoina ovat joko vastaus saatu tai aika lopussa.

3.3.5 Asynkroninen vuorovaikutus ilmoitusajastimella

Asynkronisessa vuorovaikutuksessa ilmoitusajastimella asiakasprosessi tarvitsee scope-aktiviteetin, joka sisältää invoke-aktiviteetin ja reply-aktiviteetin vastauksen hyväksymiseen onAlarm-handler-käsittelijän, joka lähettää varoituksen prosessin viivästymisestä.

3.3.6 Yksittäinen pyyntö, monta vastausta

Asiakas voi lähettää yksittäisen pyynnön palvelulle ja saada monta vastausta. Esimerkiksi pyyntö on tuotteen reaaliaikainen tilaaminen, ensimmäinen vastaus on arvioitu toimitusaika, toinen maksun vahvistaminen ja kolmas että toimitus on lähetetty.

3.3.7 Yksittäinen pyyntö, yksi vastaus kahdesta mahdollisesta

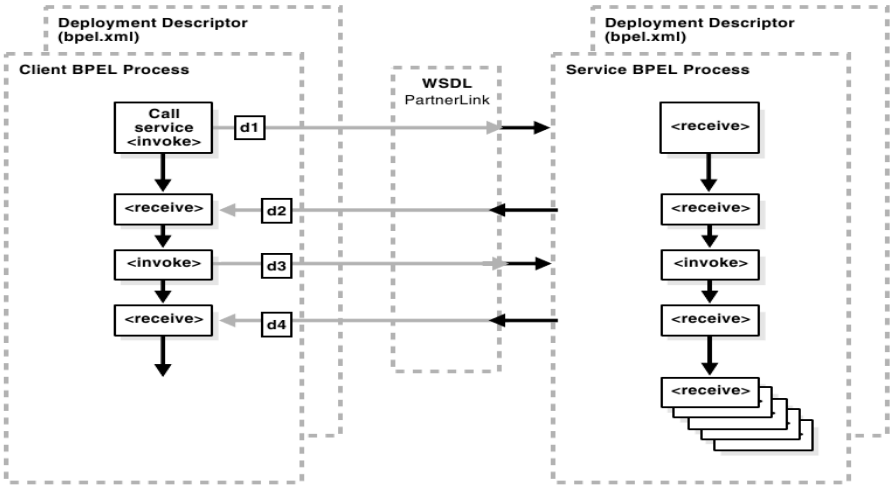
Asiakas voi lähettää yhden pyynnön palvelulle ja saada toisen kahdesta vastausvaihtoehdosta. Esimerkiksi tuotteen reaaliaikaisessa tilaamisessa asiakas lähettää yksittäisen pyynnön palvelulle. Vastauksena tulee tieto siitä, onko tuotetta varastossa vai onko varasto tyhjä.

3.3.8 Yksittäinen pyyntö, pakollinen vastaus ja optionaalinen vastaus

Asiakas voi lähettää yhden pyynnön ja saada yhden tai kaksi vastausta. Esimerkiksi tuotteen reaaliaikaisessa tilaamisessa palvelu voi lähettää aina viestin kun tuote on kuljetettu tai lähtenyt ja myöskin optionaalisesti viestin tilauksen viivästymisestä.

3.3.9 Osittainen prosessointi

Osittaisessa prosessoinnissa asiakas lähettää pyynnön ja saa vastauksen, mutta prosessointi jatkuu palvelinpuolella. Mahdollinen skenaario on esimerkiksi lomapaketin hankkiminen, jolloin palvelu lähettää välittömän vahvistuksen ja jatkaa hotellin, lennon, auton vuokrauksen jne. varaamisella. Kuva 9 havainnollistaa tätä palveluvaihtoehtoa.



Kuva 9: Osittainen prosessointi (Bradshaw et. al., 2005b)

3.3.10 Kolmas osapuoli mukana vuorovaikutuksessa

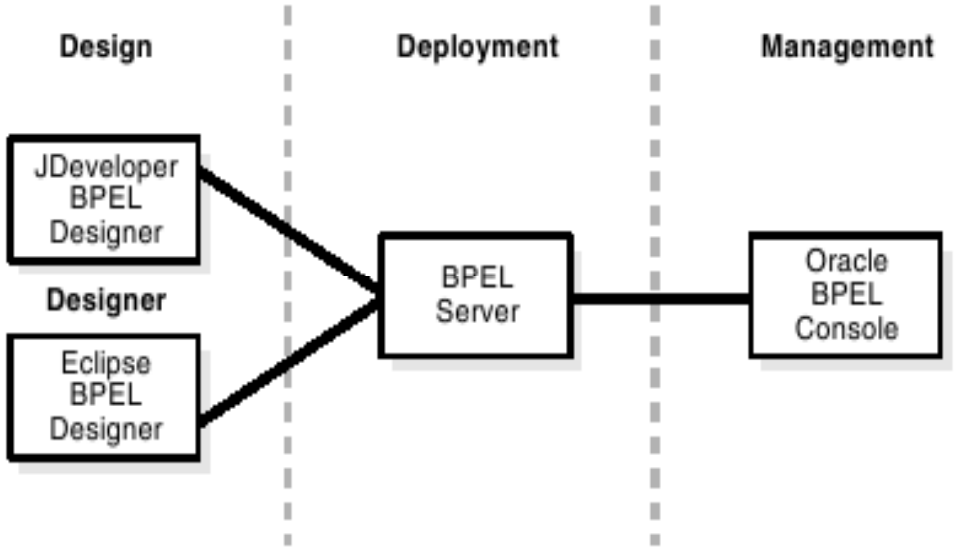
Mukana voi olla enemmän kuin kaksi sovellusta, esimerkiksi ostaja, myyjä ja huolitsija (kuljetus). Ostaja lähettää pyynnön myyjälle, myyjä lähettää pyynnön huolitsijalle ja huolitsija lähettää ilmoituksen ostajalle.

3.4 Työkalut

Luodessaan prosesseja suunnittelijan on mahdollista käyttää ainakin Oracle JDeveloper 10g, Eclipse IDE-työkalua (ks. kuva 10) tai manuaalista muokkausta. Sovellusten käyttöönotto (deploy) voi tapahtua suoraan IDE:stä käsin tai *ant*-työkalulla komentoriviltä.

3.5 Prosessien hallinta

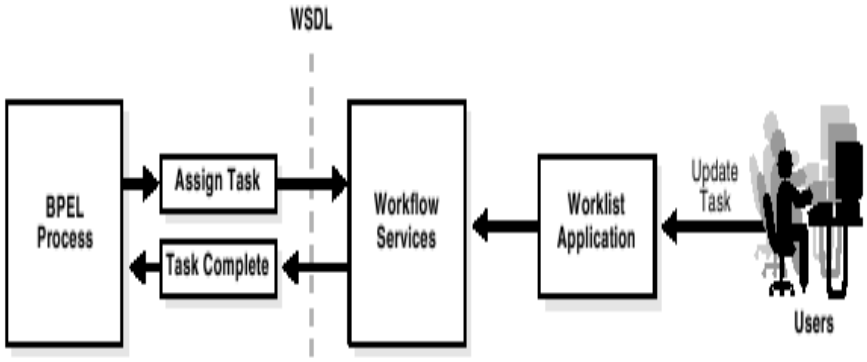
Ajonaikaisten sovellusten instanssien hallintaan Oracle on luonut Web-pohjaisen BPEL-konsolin (ks. kuva 10), jolla on mahdollista ajaa, tarkkailla ja hallita työkaluilla luotuja prosesseja (Bradshaw et al., 2005b).



Kuva 10: BPEL-komponentit (Juric et. al., 2004)

4 Työnkulkupalvelut

Työnkulkupalvelut mahdollistavat BPEL-prosessien integroimisen ihmistyön kanssa yksittäiseen prosessivirtaan. Kuvassa 11 palvelut on linkitetty BPEL-prosessiin WSDL-sopimuksen kautta. Prosessi osoittaa tehtävän tai roolin käyttäjälle ja odottaa vastinetta. Käyttäjien välineenä tehtävässä (task) on työlistasovellus (Worklist Application) (Bradshaw et al., 2005d).



Kuva 11: Työnkulkupalvelut BPEL-prosessissa (Bradshaw et. al., 2005d)

4.1 Työnkulkupalveluissa käytetyt termit

Silverin (2006) mukaan perinteiset työnkulkuohjelmistot sisälsivät aktiviteetteja tehtäviä, määräaikoja ja eskalaatioita sekä työlistoja ja tehtävien kohdentamista. Nämä ovat tyypillisiä käsitteitä pankeille, vakuutusyhtiöille ja hallituselimille, jotka ovat implementoineet työnkulkua 1980-luvun lopulta lähtien.

4.1.1 Työkulku

Työnkulku määritellään WfMC:n (1999) mukaan liiketoimintaprosessin automaatioksi, kokonaisuutena tai osittain, jonka aikana dokumentteja, tietoa ja tehtäviä (tasks)

Kuva 12: Työnkulkupalvelujen perusterminologian yhteydet (WfMC, 1999)

siirretään osalliselta toiselle proseduraalisten sääntöjen mukaisesti.

Edelleen liiketoimintaprosessin automaatio määritellään prosessimäärittelyssä, joka yksilöi erilaiset prosessiaktiviteetit, proseduraaliset säännöt ja liitetyn kontrollidatan, jota käytetään hoitamaan työnkulku prosessin toteuttamisen (enactment) aikana

Monet yksittäiset prosessi-instanssit voivat olla toiminnallisia prosessin toteuttamisen aikaan, kukin liitettynä johonkin erityiseen tietojoukkoon, joka on olennainen yksittäiselle prosessi-instanssille (tai työnkulun ”Case”:lle).

Löyhä ero vedetään joskus *tuotantotyönkulun*, jossa useimmat proseduraaliset säännöt määritellään etukäteen, ja *ad-hoc* työnkulun välille, jossa sääntöjä voidaan muokata ja luoda prosessioperaation aikana.

4.1.2 Eskalaatio

WfMC :n (1999) sanasto määrittelee *eskalaation (escalation)* proseduuriksi, automatisoiduksi tai manuaaliseksi, joka käynnistetään kun erityistä rajoitetta tai ehtoa ei saavuteta. Eskalaatioproseduurit liittyvät tavallisesti korkeampaan auktoriteettiin. Esimerkiksi jonkin tehtävän aikarajan osoitetulle käyttäjälle täyttyessä siirtyy tehtävä seuraavalle eli eskaloituu tai laajenee.

4.1.3 Tehtävät ja aktiviteetit

Kuvan 12 mukaisesti *työkohteilla (work item)* tarkoitetaan työnkulkuun osallistuville annettavia tehtäviä. WfMC (1999) määrittelee *tehtävän* aktiviteetin synonyymiksi ja edelleen aktiviteetin kuvaukseksi työn osasta, joka muodostaa loogisen askeleen prosessissa. Aktiviteetti voi olla manuaalinen aktiviteetti, joka ei tue tietokoneautomaatiota tai työnkulun (automatisoitua) aktiviteettia. Työnkulun aktiviteetti vaatii ihmisresursseja ja/tai koneresursseja prosessin suorituksen tukemiseksi. Mikäli ihmisresursseja vaaditaan, aktiviteetti kohdennetaan työnkulun osalliseksi. Käyttäjän vuorovaikutus BPEL-prosessin kanssa liittyy näin erityisesti manuaaliseen aktiviteettiin, josta WfMC toteaa (ks. kuva 12):

> Täysin manuaaliset aktiviteetit voivat muodostaa osan liiketoimintaprosessista ja tulla liitetyksi mukaan prosessimäärittelyyn, mutta ne eivät muo-

dosta osaa automatisoidusta työnkulusta johtuen tietokoneen tukemasta prosessin suorituksesta. Näin ollen aktiviteetit voidaan jaotella *"manuaalisiin"* ja *"automatisoituihin"* ja WfMC -sanastossa termi on tavallisesti käytetty viittaamaan automatisoituun aktiviteettiin.

Aktiviteettien synonyymeja ovat: askel, solmu, tehtävä, työelementti, prosessielementti, operaatio ja käsky (instruction).

4.1.4 Ilmoitukset ja tapahtumat

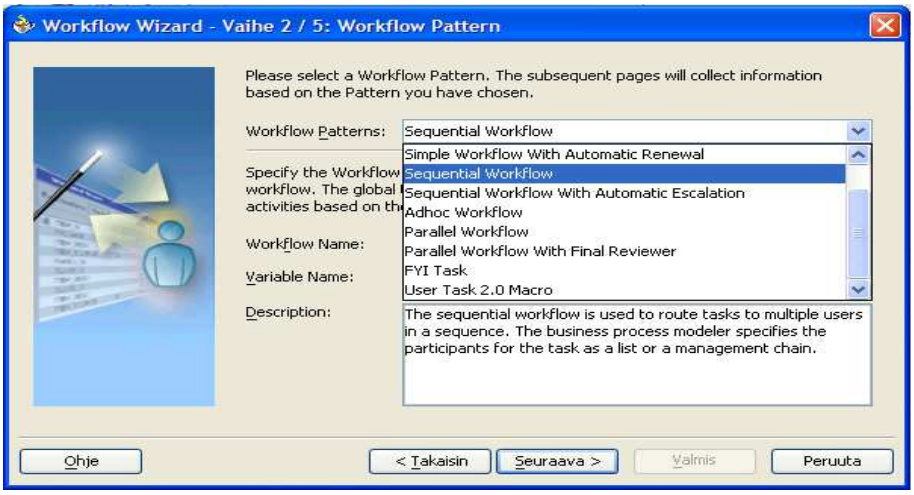
WfMC (1999) määrittelee *ilmoituksen* tapahtuman (event) synonyymiksi eli erityisen ehdon olemassaoloksi (joka voi olla ulkoinen ja sisäinen työnkulun hallintajärjestelmässä), joka aiheuttaa työnkulun hallintaohjelmiston suorittavan yhden tai useamman toimenpiteen. Esimerkiksi sähköpostiviestin saapuminen voi aiheuttaa erityisen prosessimääritysinstanssin aloittamisen. Ilmoitus (notification) - sähköposti, ääni tai tekstiviesti, joka lähetetään kun käyttäjälle on osoitettu tehtävä tai tehtävän status on muuttunut.

Tapahtumassa on kaksi elementtiä:

1. *Trigger eli laukaisin* tarkoittaa ennalta määriteltyjen olosuhteiden tunnistamista järjestelmän toiminnassa ja aiheuttaa erityisen toimenpiteen suorittamisen.
2. *Action eli toimenpide (tai response eli vaste)* on ennalta määritelty järjestelmän vaste seurauksena laukaisinehdosta.

Työnkulkujärjestelmä voi reagoida suoraan tapahtumaan tai tapahtumaa voi seurata ja tarkkailla esim. asiakasohjelma, joka alustaa toimenpiteen työnkulkujärjestelmän avulla API-kutsulla tms. Yhteys laukaisinehdon ja seuraavan järjestelmätoimenpiteen välillä edellyttää työnkulun hallintaohjelmiston käyttöä, mutta muuntyyppisiä tapahtumia voidaan käyttää sisäisesti työnkulkuun osallistuvissa sovelluksissa. Erityinen tapahtumatyyppi on eri työnkulkuinstanssien säikeiden välillä signaloiva tyyppi.

Synonyymeja tapahtumille ovat: laukaisin, toimenpide, signaali ja huomautus.



Kuva 13: Työnkulun luominen Oraclen JDeveloperin työnkulun asetusvelhossa (Ordering Book Tutorial:in mukaan). Velho käynnistyy usertask-aktiviteetin kautta. Huomaa työnkulkumallien tyyppien pudotusvalikko (Kennedy, 2006).

4.1.5 Työlista

Työlista (worklist) on luettelo tehtävistä tai työkohteista, jotka on osoitettu tai annettu tiedoksi käyttäjälle.

WfMC:n (1999) mukaan työlista muodostaa lisäksi osan rajapinnasta työnkulkumoottorin (workflow engine) ja työlistakäsittelijän (handler) välillä. Yleensä käsittelijä pyytää työn kohteita työnkulkumoottorilta muodostaakseen kyseisen listan. Joissakin työnkulunhallintajärjestelmissä työnkulkumoottori voi sijoittaa kohteet työlistaan, josta käsittelijä myöhemmin saa ne.

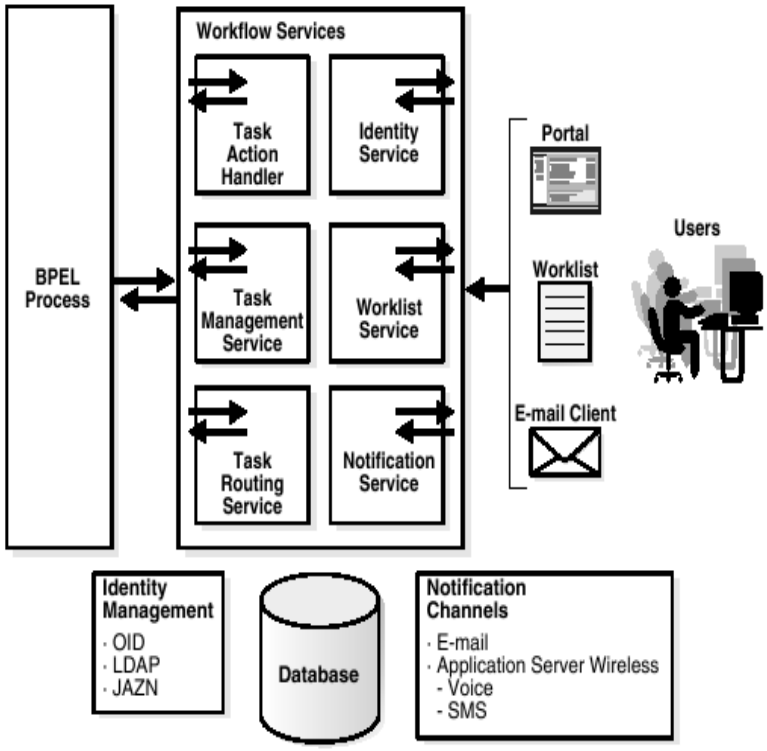
Synonyymeja työlistalle ovat TODO-listat, työjono ja In-Tray.

4.1.6 Työnkulkumalli

Työnkulkumalleja (workflow patterns) ovat: yksinkertainen hyväksyminen (simple approval), vaiheittainen hyväksyminen, samanaikainen (parallel) hyväksyminen jne. Kuva 13 havainnollistaa työnkulkumalleja. Variaatiot kuten automaattinen eskalaatio, uu-

distaminen ja muistuttajat ovat myös tuettuja. Yhdistämällä ja sovittamalla voidaan myös luoda monimutkaisia malleja.

4.2 Työnkulkupalvelun komponentteja

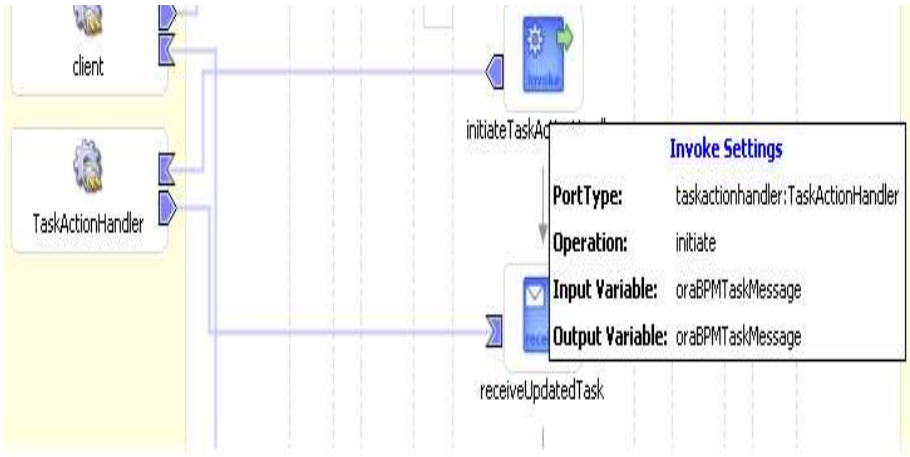


Kuva 14: Vuorovaikutus palvelujen ja liiketoimintaprosessin välillä (Bradshaw, 2005d)

4.2.1 TaskActionHandler

TaskActionHandler (ks. kuva 14) on BPEL-prosessi joka esikäynnistetään BPEL-palvelimessa. Työnkulkuprosessit käyttävät palvelua kohteiden laittamiseen käyttäjän työlistalle ja vastineiden saantiin käyttäjiltä. Prosessi myös ylläpitää ajastintapahtumia, kuten tehtävän erääntyminen, muistutus jne. (Bradshaw et al., 2005b).

TaskActionHandler-prosessi saa päivityksiä työlistasovellukselta (tai työlista-apilta) *task manager* tai *task routing* -palvelun kautta. Prosessin pääasiallinen tehtävä on pitää BPEL-prosessin tehtävä synkronoituna tietokannan tilan kanssa. Tehtävän tilaan perustuen prosessi (TaskActionHandler) päättelee kutsutaanko tehtävän BPEL-prosessi uu-



Kuva 15: Partnerilinkki TaskActionHandler-palveluun

delleen. TaskActionHandler-prosessin instanssi luodaan jokaiselle tehtävän osalliselle (Bradshaw et al., 2005a). Kuvassa 15 on esitetty TaskActionHandler-partnerilinkki.

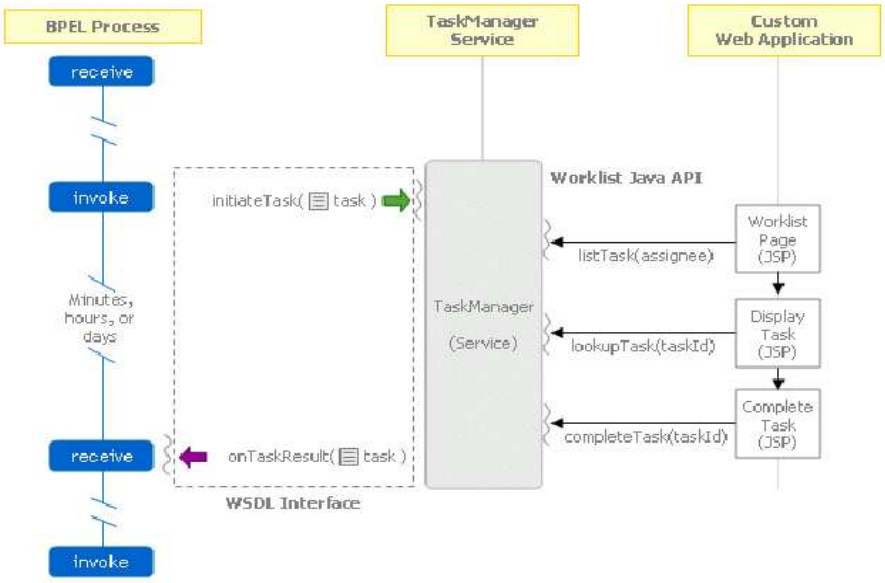
4.2.2 Tehtävienhallintapalvelu

Tehtävienhallintapalvelu (TaskManagement, TaskManager) kuvassa 16 tarjoaa tehtävän tilan hallinnan ja keston (persistence) ylläpidon sekä tehtävien päivityksen, eskalaation ja uudelleenosoittamisen. Työlistasovellus käyttää sitä tehtävien hakemiseen käyttäjille. Se myös päättää ilmoitusten lähettämisestä tehtävien tilan muuttuessa (Bradshaw et al., 2005b).

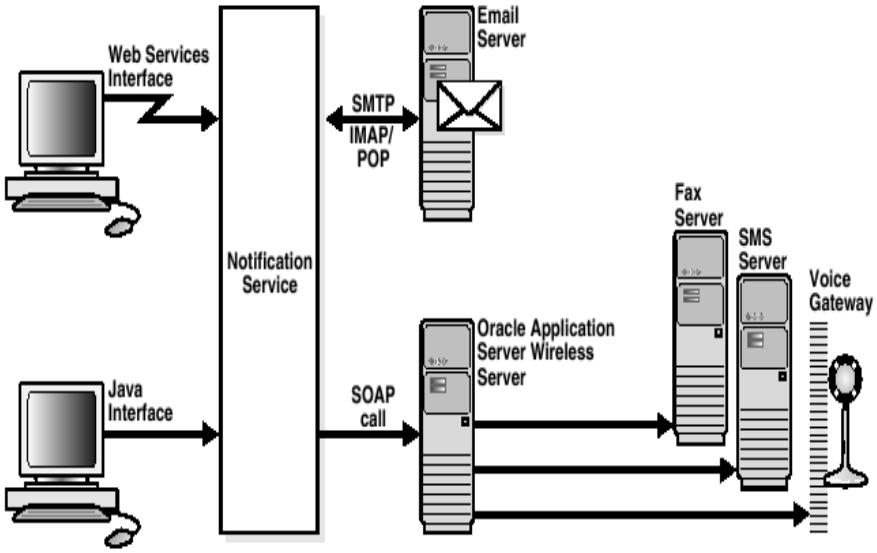
Palvelun kautta on käytettävissä mm. seuraavia operaatiota, jotka ovat saatavilla *TaskManagerService.wsdl*:n kautta WSDL-porttityyppinä *TaskManager* (Bradshaw et al., 2005a): *initiateTask, reinitiateTask, updateTask, renewTask, notifyTaskExpiration, initiateSubTask, completeTask, sendTaskReminder, releaseTask ja errorTask* .

4.2.3 Ilmoituspalvelu

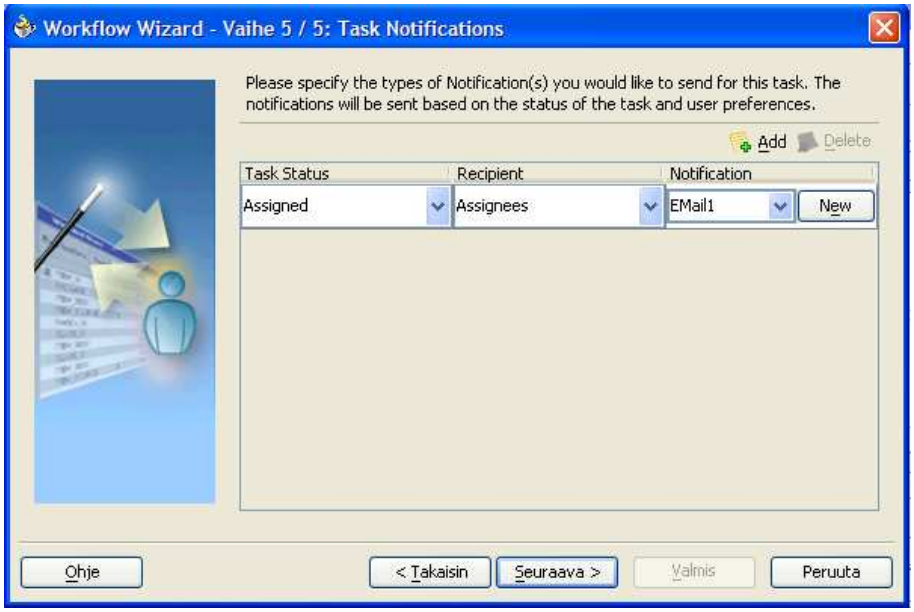
Ilmoitusaktiviteetti mahdollistaa ilmoituksen lähettämisen jostakin tapahtumasta käyttäjälle (ks. kuvat 14, 17 ja 18), ryhmälle tai sähköpostiosoiteeseen (esim.



Kuva 16: TaskManager-palvelu (Oracle, 2004)



Kuva 17: Ilmoituspalvelujen liitynnät ja tuetut palvelutyypit (Bradshaw, 2005b)



Kuva 18: Ilmoituspalvelun asetus Workflow-velholla

email,voicemail ja SMS) (Bradshaw et al., 2005b).

4.2.4 Tehtävien reitityspalvelu

Tehtävien *osoittaminen* ja *reitittäminen* sisältävät mm. (Bradshaw et al., 2005d): tehtävien luominen liiketoimintaprosessista, tehtävien osoittaminen käyttäjille ja rooleille, tuki tehtävän erääntymiseen ja automaattiseen uudistamiseen, tuki tehtävän delegoimiseen, laajentamiseen ja uudelleenhyväksymiseen.

Reitityspalvelut (Task Routing Service) (ks. kuva 14) ovat saatavilla *TaskManager* -palvelun kautta WSDL-porttityyppinä, jonka nimi on *TaskRoutingService.*

Reitityspalvelun operaatiot ovat: *routeTask, routeTaskToNextApprover, escalateTask ja escalateTaskOnExpiration* (Bradshaw et al., 2005a).

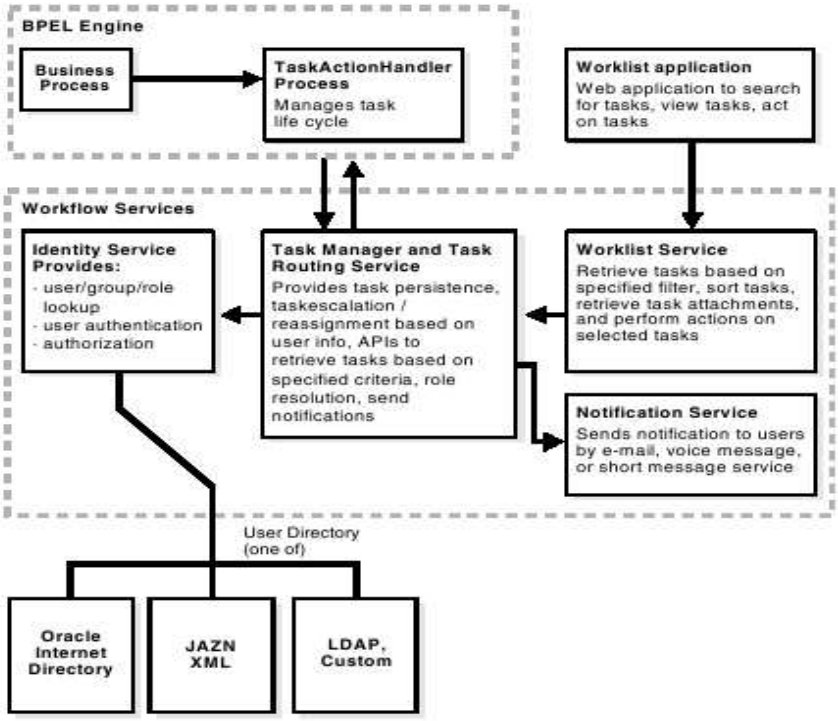
4.2.5 Tunnistamispalvelu

Tunnistamispalvelu (identity service) on vuorovaikutuksessa identiteetin hallintasysteemien, kuten Java AuthoriZatioN (JAZN) ja LDAP, kanssa. Identiteettipalvelut tarjoavat roolipohjaisen pääsynhallinnan (access control) eli oikeuksia voidaan osoittaa

rooleille ja yhdistää organisaationaalinen hierarkia autorisoituun roolimalliin. Lisäksi voidaan osoittaa työlistaoikeuksia käyttäjille, rooleille ja ryhmille, pitää yllä käyttäjäominaisuuksia kuten nimi, sijainti, puhelin, faksi ja sähköposti, koota organisaation hierarkia- ja ryhmätietoa ja integroida käyttäjä standardiin hakemistopalveluun (esim LDAP) (Bradshaw et al., 2005d).

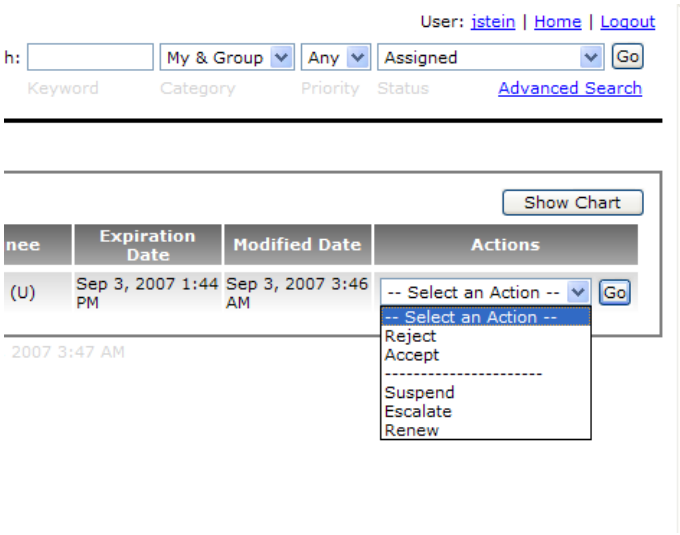
5 Työlistasovellus ja työlistapalvelun ohjelmointirajapinta

Työlistasovellus (Worklist Application) on työlistapalvelun rajapinnan (Worklist API) avulla rakennettu web-pohjainen BPEL-esimerkkisovellus, jolla on kyky hakea tehtäviä valikoidusti ja suorittaa tehtävään liitettyjä toimenpiteitä. Se antaa mahdollisuuden osallistua BPEL-prosessiin suorittamalla tehtäviä, jotka vaativat manuaalista puuttumista. Kuvassa 19 on esitetty kuinka työlistasovellus liittyy työnkulkupalveluihin.



Kuva 19: Vuorovaikutus työvirtapalvelujen ja liiketoimintaprosessien välillä (Bradshaw, 2005d)

Työlistasovelluksen käyttöliittymä näyttää käyttäjäkohtaiset tehtävät kirjautuneelle käyttäjälle, tehtävät jotka perustuvat käyttäjän oikeuksiin, ryhmiin ja rooleihin (Bradshaw et al., 2005d). Haluttaessa hienosäädetympää käyttöliittymää voidaan webliittymä toteuttaa servlettien ja JSP-sivujen avulla. Tavallisia toimenpiteitä, joita sovelluksella voi suorittaa, ovat ensin kirjautua sovellukseen, tarkastella tehtäviä ja sitten esim. hyötykuorman (payload) päivittäminen, dokumenttien ja kommenttien liittä-

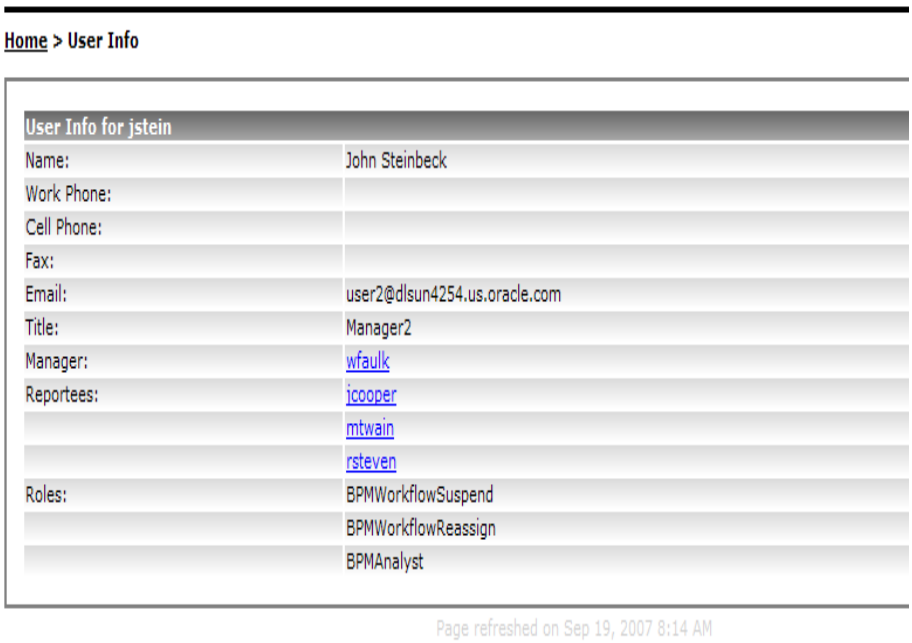


Kuva 20: Työlistasovelluksen web-liittymä ja toimenpiteen valinta

minen, tehtävän reitittäminen toiselle käyttäjälle ja tehtävän päättäminen, kuten hyväksyminen tai hylkääminen. Kuvassa 20 on eritelty tehtävään kohdistuvan toimenpiteen valinta.

5.1 Työlistasovelluksen käsitteiden yleiskuvaus

Esimerkkinä työlistasovelluksen käyttötapauksista on kehittäjän oppaassa (Bradshaw et al., 2005d) mainittu tehtävät: esimiehen täytyy hyväksyä työntekijän lomapyyntö tai pankkivirkailijan täytyy tarkastaa lainahakemus, jotka on toimitettu osana BPEL-prosessia.



Kuva 21: Käyttäjäinfo

Tehtäviin kohdistuvia toimenpidetyyppejä ovat mm:

- *Päivittäminen* - sisällön muokkaaminen ja lisääminen. flex-kenttien[1] muokkaaminen
- *Päätöksen muuttaminen* - jos näin tehdään, tehtävä poistetaan työlistalta ja reitätään seuraavalle hyväksyjälle tai liiketoimintaprosessiin työvirtamallin mukaisesti.
- *Järjestelmätoimenpiteet* - toimenpiteet ovat käytettävissä kaikille tehtäville käyttäjän oikeuksien perusteella. Toimenpiteet ovat:
  - *Eskalointi (escalate)* - käyttäjä voi eskaloida tehtävän johtajalleen.
  - *Uudelleenosoitus (reassign)* - riittävin oikeuksin tehtävän voi delegoida.

[1] Tehtäväobjekti sisältää flex-kenttiä tehtävän laajentamiseen lisädatan kiinnittämiseen payload-datan lisäksi. Flex-kenttiä kohdellaan samalla tavalla kuin muita otsikkoattribuutteja työlistalla. Tehtäviä voidaan etsiä tämän datan perusteella. Flex-kentät ovat saatavilla seuraaville datatyypeille:*string, double, long ja date* (Bradshaw et al., 2005d)

- *Lisätiedon pyytäminen* - jokainen osallinen työnkulussa voi pyytää lisätietoa tehtävän luojalta tai edellisiltä hyväksyjiltä. Kun pyydetty tieto toimitetaan, tehtävä osoitetaan pyytäjälle.
- *Lisätiedon lisääminen* - mahdollistaa käyttäjän vastata lisätiedon pyytämiseen. Tehdään sen jälkeen kun käyttäjä on lisännyt tarpeelliset päivitykset tehtävään ja lisännyt kommentit ja liitteet, jotka sisältävät lisätiedon.
- *Reititys (route)* - esim. päättäminen ja reititys muille tarkasteltavaksi.
- *Pysäyttäminen (suspend)* - tehtävän väliaikanen pysäyttäminen sen omistajan tai *BPMWorkflowSuspend* oikeuksin (ks. kuva 21). Tehtävä ei pääty tai eskaloidu ennen sen uudistamista.
- *Palauttaminen (resume)* - em. oikeuksin pysäytetyn tehtävän palauttaminen.
- *Hankinta (acquire)* - antaa käyttäjälle erikoisoikeuden työskennellä ryhmälle tai useammalle käyttäjälle osoitetun tehtävän kanssa.
- *Irrottaminen (release)* - irtautuminen yksinoikeudesta, jolloin muilla käyttäjillä mahdollista hankkia yksinoikeus.
- *Vetäytyminen (withdraw)* - tehtävän luoja tai omistaja vetää vireillä olevan tehtävän pois.

5.1.1 Työlistapalvelun rajapinta ja TaskManager

Kuvassa 16 on esitetty TaskManagerin rajapinnat BPEL-prosessin (WSDL-rajapinta) ja kustomoidun JSP-sivuston (Worklist Java API) kanssa.

Tavallisesti askeleet integroitaessa TaskManager-palvelu BPEL-prosessiin ovat (Bradshaw et al., 2005a):

- Partnerilinkin määrittäminen TaskManager-palveluun.
- Tehtävädokumentin (XML-dokumentti)[2] julkistaminen (declare) ja alustaminen.

[2]BPEL-prosessissa kaikki data on XML-formaatissa kuten viestit BPEL prosessista, viestit jotka vaihdetaan ulkoisten palvelujen kanssa ja paikalliset muuttujat joita tietovirta käyttää. Viestien tyypit määritellään XML-skeemalla, tavallisesti WSDL-tiedostolla virtaa varten tai WSDL-tiedostoilla palveluille, joita skeema käynnistää. Niinpä kaikki muuttujat ovat XML-dataa ja kaikki BPEL prosessit käyttävät suuren osan koodistaan niiden muuntamiseen.

- Invoke-aktiviteettia käyttäen TaskManagerin *initiateTask* -operaation käynnistäminen kuten mikä tahansa web-palvelu.
- Receive-aktiviteetilla *onTaskResult*-takaisinkutsun (callback) odottaminen TaskManager-palvelulta.
- Päivitetyn tehtävädokumentin lukeminen takaisinkutsuviestistä.

Askeleet tehtävän tietoihin pääsemiseksi käyttäjän puolelta ovat (ks. kuva 16):

- Graafinen käyttöliittymä käyttää Java Worklist API:a listaamaan tehtävät, jotka on osoitettu käyttäjälle tai roolille.
- Käyttäjä valitsee tehtävän ja käy läpi yksityiskohtia.
- Käyttäjä päivittää editoitavaa tehtävän dataa ja tallentaa tai päättää tehtävän.

Kohdassa 6.3.5 BPEL-prosessi, johon on liitetty perinnesovelluksesta luotu palvelu, on osa yksinkertaista työnkulkupalvelua, jossa Oracle:n työlistasovelluksen avulla käyttäjä käy läpi annetun tehtävän.

6 Perinnesovelluksesta webpalvelu

Vanhoihin sovelluksiin liittyy käyttäjän vanhaa osaamista ja siitä syystä on tarpeen liittää näitä esimerkiksi webpalvelujen kautta osaksi uusia sovelluksia tai palveluita. Tämän luvun esimerkki luo web-palvelun MySQL-tietokannan rajapinnan avulla. Vaikka tietokanta ei ole perinnesovellus, niin kannan C-rajapinta tarjoaa esimerkin esittämiselle perinnesovelluksiin sopivan kehyksen.

6.1 SOAP-protokolla

SOAP eli *Simple Object Access Protocol* on XML-pohjainen kommunikaatioprotokolla, joka on tarkoitettu sovellusten väliseen viestien lähettämiseen, toimii ineternetprotokollien avulla ja on sekä kieli- että käyttöjärjestelmäriippumaton. RPC:hen ja CORBA:an verrattuna internetprotokollat mahdollistavat helpomman läpaisevyyden palomuureista (W3C, 2000).

6.2 gSOAP

gSOAP-työkalut mahdollistavat SOAP/XML-to-C/C++ kielisidonnat helpottaen siten SOAP/XML web-palvelujen ja asiakasohjelmien kehittämistä C:llä ja C++:lla (Engelen, 2008). Tarvittaessa gSOAP-työkalut (gSOAP Web Services Toolkit) mahdollistavat sovellusdatan eli C/C++ -tietorakenteiden sarjallistamisen (serialization) XML:ssä. Työkalut sisältävät lähdekoodigeneraattorin, WSDL ja XML-skeema analysaattorit skeematyyppien sitomiseksi suoraan C/C++ -datatyyppeihin (Engelen, 2005).

gSOAP:n kehittäjän Robert Engelenin mukaan useimmat työkalut C++ web-palveluille soveltavat SOAP-keskittynyttä näkökulmaa ja tarjoavat API:ja, jotka vaativat luokkakirjastojen käyttöä SOAP-spesifisissä tietorakenteissa. Tämä pakottaa usein kehittäjän/käyttäjän sopeutumaan näiden kirjastojen sovelluslogiikkaan. gSOAP tarjoaa tähän ratkaisuksi läpinäkyvän C/C++ SOAP API:n, joka käyttää kääntäjäteknologiaa SOAP-spesifisten yksityiskohtien peittämiseen. Käytännössä gSOAP-kääntäjä (luo tyngät (stubs) ja rungon (skeleton)) kartoittaa automaattisesti natiivit ja käyttäjän määrittämät C ja C++ tietotyypit semanttisesti yhtäpitäviin XML-tietotyyppeihin ja päinvastoin (Engelen, 2008).

Sovelluksen rakentaminen tapahtuu käytännössä joko luomalla valmiista WSDL-tiedostosta wsdl2h-jäsentäjällä otsikkotiedoston ja sitten soapcpp2 -kääntäjällä luodaan runko C - tai C++ -kielisenä sovelluksen rakentamista varten. Jäsennysvaiheen voi myös ohittaa ja tehdä tarvittaessa otsikkotiedoston manuaalisesti, jolloin kääntäjä luo tarvittavan WSDL-tiedoston.

6.3 SOAP-palvelun rakentaminen gSOAP:n avulla

Seuraavassa on rakennettu Linux-alustalle yksinkertainen web-sovellus rakentamalla ensin SOAP-protokollaa käyttävä C++ -sovellus ja gsoap-kääntäjällä muodostettu web-palvelu. Sovellus hakee tietokannasta jäsenluettelon ja palauttaa sen merkkijonona, jossa jäsenet on eroteltu toisistaan pilkuilla. SOAP-palvelu toimii Apache2-webpalvelimen moduulina (ks. kuva 28) ja toiselle webpalvelimelle Oraclen sovelluspalvelimella gSOAP-palvelun WSDL-tiedostosta luodaan JDeveloperissa partnerilinkki, jonka avulla sitten haetaan jäsenlista hyötykuormana ja kommenttiliitteenä ”käyttäjän liittymään” eli *Oracle BPEL Worklist Application*:iin käyttäen usertask-aktiviteettia yksinkertaisella työnkulkumallilla (ks. kuva 13).

6.3.1 Tietokantamoduuli

Kuvassa 22 on esitetty C-kielinen moduuli, joka palauttaa tietokannasta jäsenlistan, jossa nimet ovat eroteltuina pilkuilla.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <mysql/mysql.h>

char *  mysli(char *nimi, char *sana){
MYSQL *mysqlcontext;
MYSQL_RES *mysqlres;
MYSQL_ROW row;
int count,n;
char *point,*p,*np;

    mysqlcontext = mysql_init(NULL);
    if (!mysql_real_connect(mysqlcontext,"localhost",nimi,sana,
                            "jasenrek",0,NULL,0))
     {
          fprintf(stderr,"%s\n",mysql_error(mysqlcontext));
          return "";
     }
    if (mysql_query(mysqlcontext,"select * from jr_jasen"))
    {
        fprintf(stderr, "%s\n",mysql_error(mysqlcontext));
        return "";
    }
    mysqlres = mysql_use_result(mysqlcontext);
    count = 0;
    point = malloc(1000);
    while((row = mysql_fetch_row(mysqlres)) != NULL)
    {
  point = strcat(point,row[1]);
          point = strcat(point," ");
          point = strcat(point,row[2]);
          point = strcat(point,",");
     }
    point = strcat(point,"\n\n");
    mysql_free_result(mysqlres);
    mysql_close(mysqlcontext);
 return  point+5;
 }
```

Kuva 22: C-moduuli MySql-tietokantahakua varten.

Moduulin kääntäminen objektitiedostoksi käy seuraavalla komennolla:

```
> gcc -c mysli.c
```

6.3.2 C++-osa

Kuvan 22 funktion kutsu liitetään C++-metodiin kuvan 23 esittämällä tavalla. Koodiin on sisällytetty kuvan 22 funktio *char∗ mysli()* jota kutsutaan *char ∗ JasenlistaLibrary::SayMySQL()*-metodin kautta.

```
#include "JasenlistaLibrary.h"
#include <string.h>
#include <iostream.h>
extern "C"{
char * mysli(char*,char*);
}
JasenlistaLibrary::JasenlistaLibrary(){}

JasenlistaLibrary::~JasenlistaLibrary(){}

char* JasenlistaLibrary::SayMysql(char *nimi,char *sana)
{
 return mysli(nimi,sana);
}
```

Kuva 23: JasenlistaLibrary.cpp.

6.3.3 Otsikkotiedosto skeleton-kääntäjälle

Aluksi teemme soap-rungon muodostamista varten kuvassa 24 esitetyn otsikkotiedoston ”jasenlista.h”.

```
// Content of file "jasenlista.h"
//gsoap ns service location: http://localhost.localdomain/cgi-bin/jasenlista.cgi
int ns__jasenlista(char *nimi,char *sana,char *&result);
```

Kuva 24: Otsikkotiedosto

josta muodostamme skeleton-tiedostot soapcpp2-kääntäjällä kuvan 25 mukaisesti.

```
[mh@localhost gsoap]$ make wsdl
/usr/local/bin/soapcpp2 -t inc/jasenlista.h

**  The gSOAP Stub and Skeleton Compiler for C and C++ 2.7.10
**  Copyright (C) 2000-2008, Robert van Engelen, Genivia Inc.
**  All Rights Reserved. This product is provided "as is", without any warranty.
**  The gSOAP compiler is released under one of the following three licenses:
**  GPL, the gSOAP public license, or the commercial license by Genivia Inc.

Saving soapStub.h
Saving soapH.h
Saving soapC.cpp
Saving soapClient.cpp
Saving soapClientLib.cpp
Saving soapServer.cpp
Saving soapServerLib.cpp
Using ns service name: Service
Using ns service style: document
Using ns service encoding: literal
Using ns service location: http://localhost.localdomain/cgi-bin/jasenlista.cgi
Using ns schema namespace: http://tempuri.org/ns.xsd
Saving ns.wsdl Web Service description
Saving soapProxy.h client proxy
Saving soapObject.h server object
Saving ns.jasenlista.req.xml sample SOAP/XML request
Saving ns.jasenlista.res.xml sample SOAP/XML response
Saving ns.xsd XML schema
Saving ns.nsmap namespace mapping table

Compilation successful
```

Kuva 25: Skeleton-tiedoston muodostaminen

Kuvan 25 listauksesta olennaisin on web-palvelua kuvaava tiedosto ns.wsdl, jota myöhemmin käytetään partneri-linkin luomiseen.

Kääntäjän luomaa runkoa voi käyttää myös web-palvelimesta erillään toimivan *standalone* tyyppisen palvelun ja *proxy*-tyyppisen asiakasohjelman tekemiseen. Asiakasohjelma on kokoelma luokkia, jotka yhdessä muodostavat ja käsittelevät SOAP-viestejä asiakkaan puolella. Proxyn tarkoitus on piilottaa yksityiskohdat SOAP-prosessoinnista ja http- tai https-kuljetuksesta. Tällaista asiakasohjelmaa voi käyttää esimerkiksi web-palveluja käyttävissä mobiililaitteissa. Kuvassa 25 on skeleton-kääntäjän muodostama proxy-asiakkaan otsikkotiedosto *soapProxy.h*.

6.3.4 Palvelin-osa

Kuvan 26 Server.cpp käynnistää palvelun. Koodin main-funktiossa *soap_serve(soap_new())*-osan avulla käynnistetään gSOAP-palvelinprosessi. Kuvan 24 otsikkotiedostossa soapcpp2-kääntäjälle annettu *ns_jasenlista()*-funktio kutsuu ensin kuvan 23 *SayMysql*-metodia, joka puolestaan kutsumalla kuvan 22 *mysli()*-funktiota hakee jäsenlistan. Kuvassa 24 esiteltyyn *ns_jasenlista()*-funktioon pitää lisätä ”ylimääräinen” *struct soap *soap*-parametri vastaavassa kuvan 26 *ns__jasenlista()*-funktiossa.

```
#include "soapH.h"
#include "ns.nsmap"
#include "JasenlistaLibrary.h"

int main()
{
struct soap soap;
soap_init(&soap);
//soap_set_namespaces(&soap,namespaces);
soap_serve(soap_new());
}

int ns__jasenlista(struct soap *soap,
                   char *nimi,
                   char *sana,
                   char *&result)
{
JasenlistaLibrary jasenlista;

result = jasenlista.SayMysql(nimi,sana);

return SOAP_OK;
}
```

Kuva 26: Server.cpp.

Lopuksi käännämme rungon, tyngät (stub), lähdekoodin ja objektin *jasenlista.cgi*-tiedostoksi, joka on valmis ajettavaksi palveluna Apachen gSOAP-moduulissa. Kään-

nös tapahtuu komentorivillä seuraavasti:

```
g++ -o jasenlista.cgi muut/mysli.o soapC.cpp soapServer.cpp \
src/Server.cpp -I. -I./inc -L/usr/local/lib -lgsoap++ -L. \
-lJasenlistaLibrary -L/usr/lib64/mysql -lmysqlclient
```

Tuloksena saatu jasenlista.cgi kopioidaan tässä tapauksessa Apachen gSOAP-moduulia varten /var/www/cgi-bin -hakemistoon. gSOAP-moduulin kääntämiseen lähdekoodista käytetään apxs eli APache eXtenSion tool -työkalua ohjeen mukaisesti seuraavasti:

```
Here's how to compile mod_gsoap for apache 2.0:

/usr/local/apache2/bin/apxs -a -i -c \
-I/path/to/gsoap/installation mod_gsoap.c
```

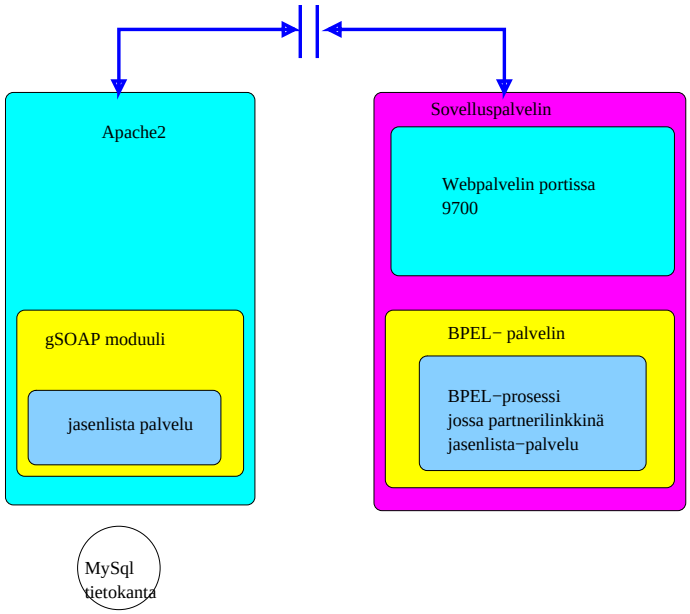
Edellisen toimenpiteen hakemiston sisältö on esitelty kuvassa 27.

```
[mh@localhost apache_20]$ pwd
/home/mh/gsoap-2.7/gsoap/mod_gsoap/mod_gsoap-0.6/apache_20
[mh@localhost apache_20]$ ls
apache_gsoap.h  mod_gsoap.la  mod_gsoap.o    mod_gsoap.vcproj
mod_gsoap.c     mod_gsoap.lo  mod_gsoap.slo  README.txt
```

Kuva 27: gSOAP:n Apache-moduulia varten tarvittavat tiedostot.

Moduulin lataamiseksi tarvitaan seuraava rivi Apachen kokoonpanotiedostoon ja serverin uudelleenkäynnistys:

```
LoadModule gsoap_module       /usr/lib64/httpd/modules/mod_gsoap.so
```



Kuva 28: SOAP-webpalvelu ja partnerlinkki-asiakas

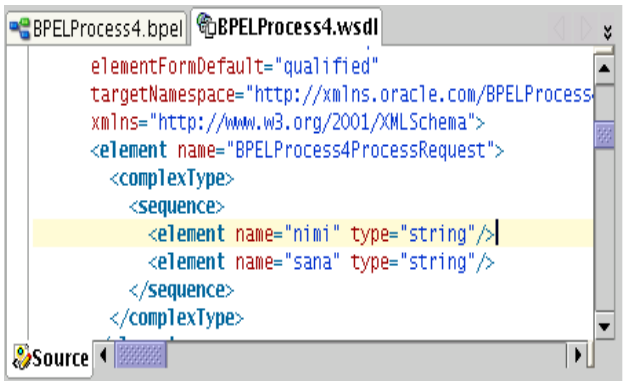
6.3.5 Integrointi partnerilinkiksi BPEL-prosessiin

Liitettäessä gSOAP:lla luotu palvelu BPEL-prosessiin tarvitaan joitakin ylimääräisiä toimenpiteitä, kuten BPEL-prosessin WSDL-tiedoston muokkausta (kuva 29) yhteensopivaksi gSOAP-palvelun kanssa ja assign-aktiviteetin käyttöä parametrien viennissä palveluun ja myös palautettaessa palvelun kautta saatu merkkijono BPEL-prosessiin.

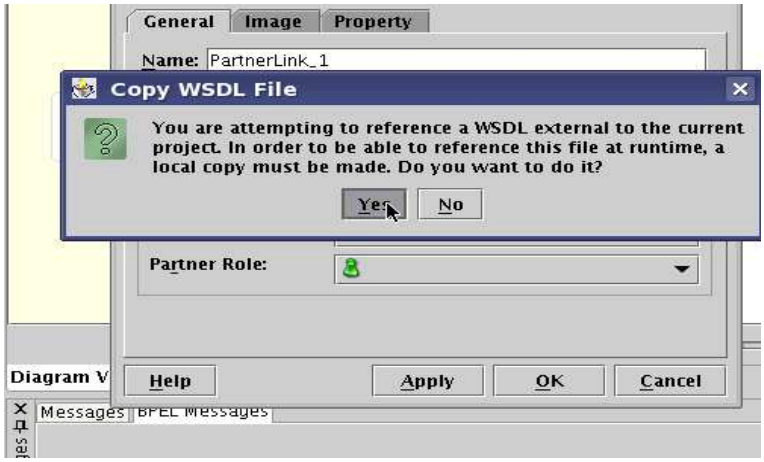
Tähän työlistatyyppiseen BPEL-sovellukseen palvelua liitettäessä vaiheet ovat:

1. *Partnerilinkin luominen* partnerlink-aktiviteetin avulla.
2. *Partnerilinkin invokaation* luominen invoke-aktiviteetin avulla.
3. *Käyttäjän tehtävän luominen* usertask-aktiviteetin avulla. Usertask-aktiviteetti luo käytännössä joukon muita aktiviteetteja, jotka luovat yhteydet tehtävienhallinta-, tunnistuspalvelu- ym. partnerilinkkeihin. Usertask-aktiviteettia ei näy prosessisymbolina sen asettamisen jälkeen.

Kun gSOAP:lla luotu wsdl-tiedosto ”ns.wsdl” viedään partnerilinkkiin (kuva 31), ohjelma luo sovittavan ”nsRef.wsdl” tiedoston, johon luodaan porttien oletustyypit partnerilinkkiin (kuva 32).

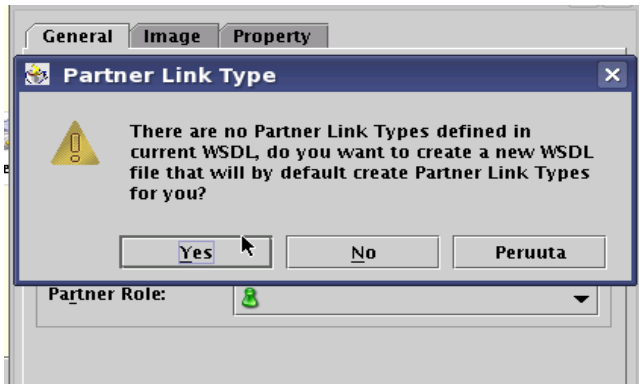


Kuva 29: Muutetut parametrit BPEL-prosessin WSDL-tiedostossa. Alkuperäisen yhden merkkijonon parametrin tilalla on kahden merkkijonon parametrit ”nimi” ja ”sana”, joiden avulla BPEL-konsolista käsin prosessi käynnistetään.

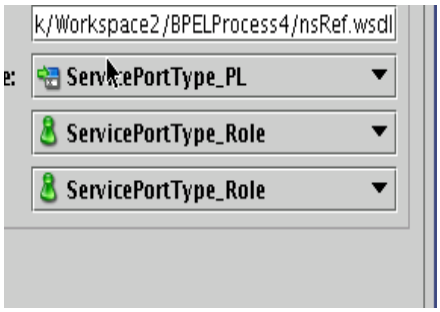


Kuva 30: gSOAP:lla luodun WSDL-tiedoston tuonti partnerilinkkipalveluksi

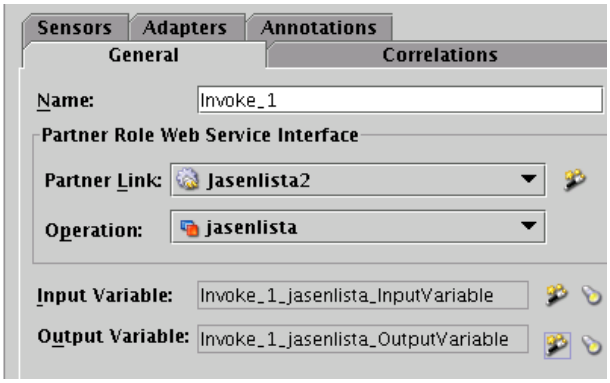
Kuvassa 25 luodun ns.wsdl:n kuvaaman palvelun tuonti partnerilinkiksi on havainnollistettu kuvassa 30.



Kuva 31: Partnerilinkiksi luominen.

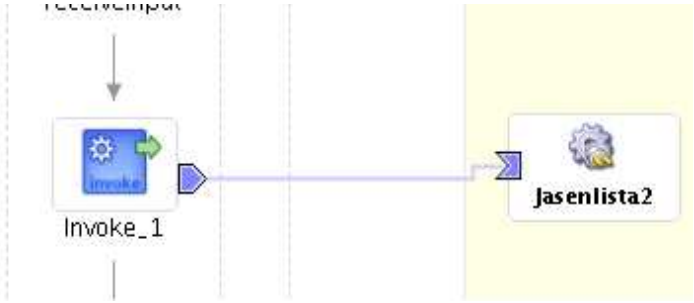


Kuva 32: Porttityypin valinta



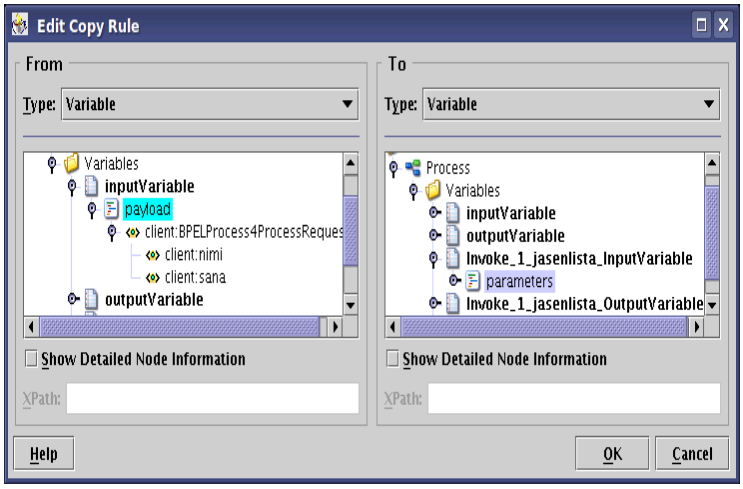
Kuva 33: Partnerilinkin invokaation määrittely

Kuvan 33 mukaisesti invoke-aktiviteettiin valitaan jasenlistafunktio ja luodaan syöte- ja tulostemuuttujat.



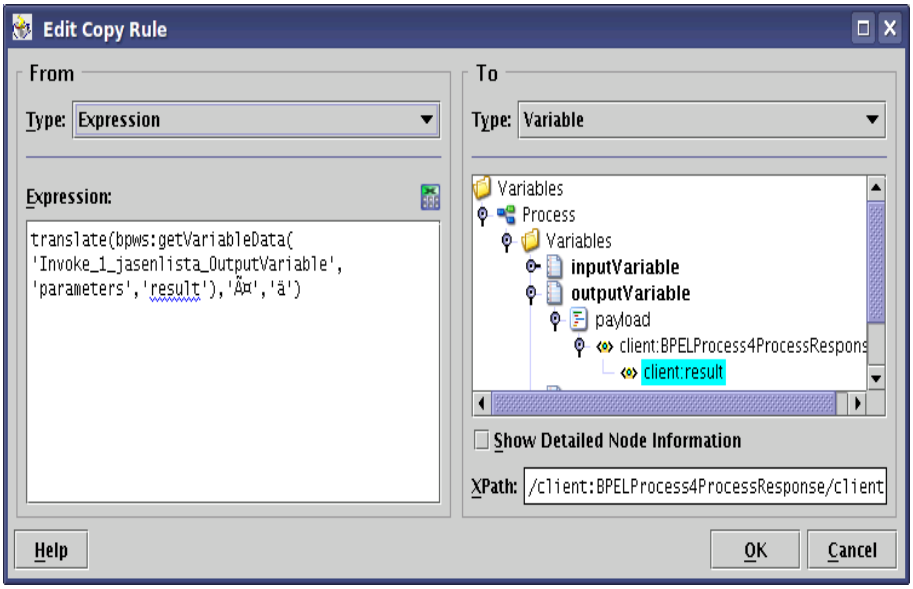
Kuva 34: Invoke-aktiviteetti ja jäsenlistan partnerilinkki.

Kuvan 32 määrittelyn seurauksena täydentyy prosessikaavio kuvan 34 ilmaisemalla tavalla yhdellä invoke-symbolilla ja yhdellä partnerilinkki-symbolilla (ks. myös kuva 28).



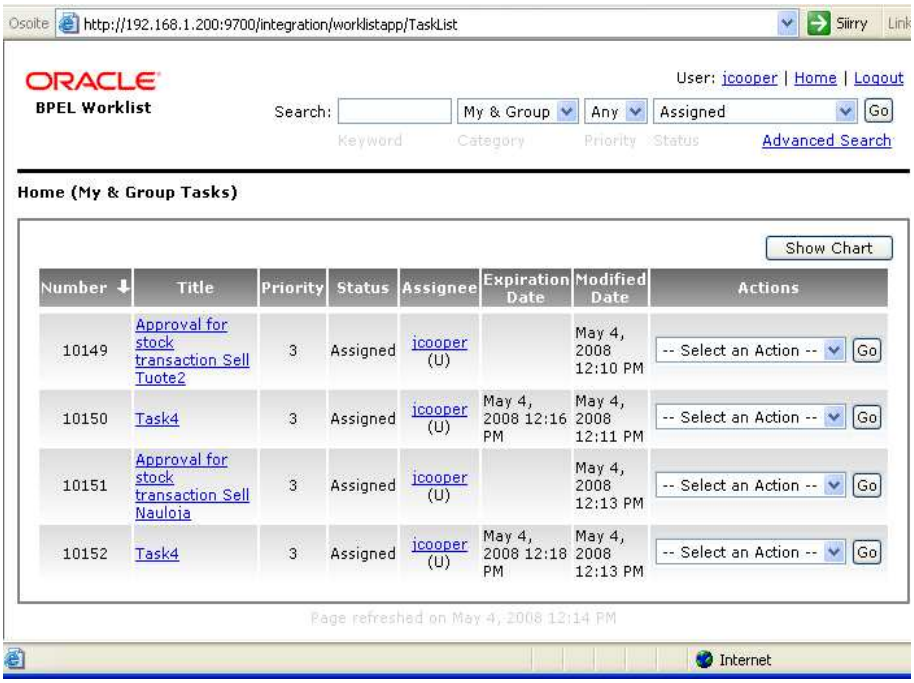
Kuva 35: Parametrien siirto jasenlista-partnerilinkin invokaatioon.

Kuvassa 35 on esitetty, kuinka kuvan 29 parametrit ”nimi” ja ”sana” (eli MySQL-kannan käyttäjätunnus ja salasana) siirretään prosessivirrasta partnerilinkin parametreiksi.



Kuva 36: Webpalvelusta saadun merkkijonon siirto prosessivirtaan

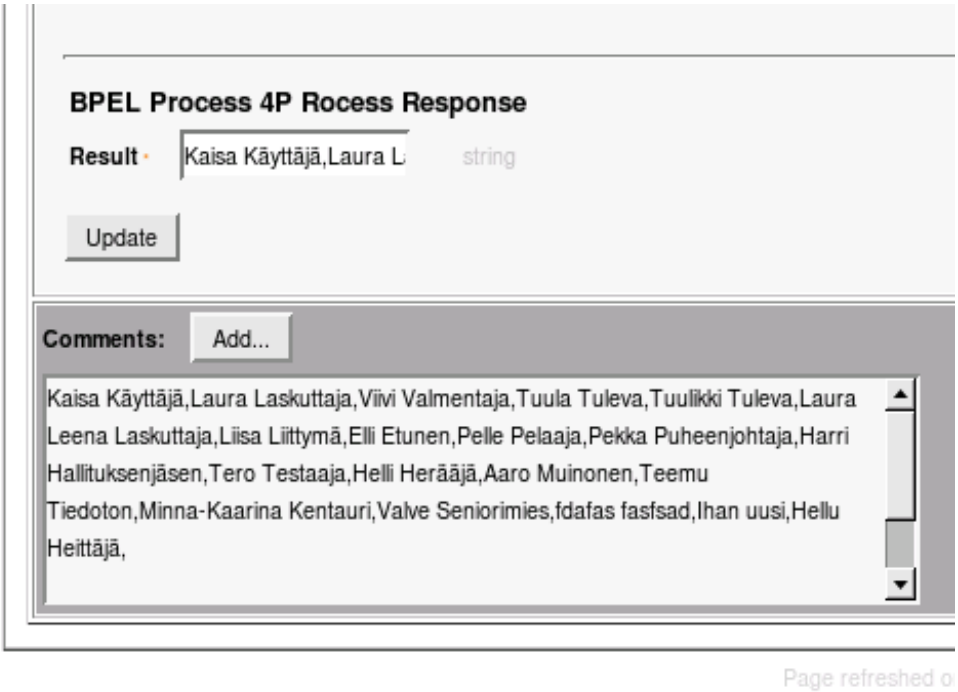
Kuvassa 36 siirretään assign-aktiviteetilla invoke-aktiviteetista saatu jasenlista merkkijonona prosessivirran tulostemuuttujan rakenteeseen. Samalla muunnetaan koodistoa yhteensopivaksi.



Kuva 37: Työlista ”Oracle BPEL Worklist” -sovelluksessa

Kuvassa 37 on työlista-API:a varten tehty Oraclen esimerkkisovellus, jossa näkyvät käyttäjän ”jcooper” hyväksymistä työlistalla odottavat tehtävät. Yhteys työlistaan ja sen perustana oleviin, luvuissa 4 ja 5 käsiteltyihin, työnkulkupalvelun muodostaviin BPEL-prosesseihin muodostetaan esimerkiksi usertask-aktiviteetin avulla.

Kuva 38: Lopputulos palvelusta ”Oracle BPEL Worklist” -sovelluksessa.



Kuva 39: Jäsenlista lisättynä kommentteihin.

Kuvassa 38 on lopputulos työlista-sovelluksessa. Web-palveluna oleva jäsenlistan palauttavan partnerilinkin tuotos on BPEL-prosessin käyttämänä siirtynyt halutulle käyttäjälle. Prosessin ”hyötykuormana” ollut jäsenlista-merkkijono on kulkenut tunnistus-, tehtävänhallinta-, reititys-, työlistapalvelun ja *TaskAction*-käsittelijän avulla käyttäjän työlistalle, josta ilmoituspalvelun avulla käyttäjä ”jcooper” on saanut tiedon odottavasta tehtävästä. Kuvassa 39 sama result-kentän merkkijono on assign-aktiviteetin avulla siirretty käyttäjälle havainnollisempaan kommenttikenttään.

6.3.6 gSOAP:n rajoituksia

Rajoituksia gSOAP:ssa ovat mm. (Engelen, 2008):

- C/C++ -kielen näkökulmasta, joitakin C++ kielen ominaisuuksia ei voida tukea SOAP:n ulkoisten (remote) metodien määrittelyssä.
- Määrätyt rajoitukset C++ kielen rakenteissa:

- Mallien (templates) instansseja ei voida käyttää main-osassa ohjelmaa, koska gSOAP-kääntäjä on esiprosessori, joka ei voi päätellä instansseja eikä kehittää mallien koodia.
- Moniperintää ei voida tukea SOAP-protokollan vuoksi.
- Abstraktit metodit: luokan täytyy olla instantioitavissa, että luokkien instanssien dekoodaus toimisi.
- gSOAP-kääntäjän tulkinta direktiiveistä ja pragmoista, kuten *#include*:n ja *#define*:n poikkeaa tavanomaisesta käsittelystä.
- Kaikki luokkien metodit luokissa pitää julkistaa (declare) luokan otsikkotiedostossa, mutta metodeja ei pitäisi käyttää koodissa, vaan metodien implementaatiot täytyy määritellä erillisessä C++- tai C-lähdetiedostossa ja linkittää se sovellukseen.

7 Yhteenveto

BPEL:in viitekehys on sijoitettavissa samalle alueelle kuin EAI ja ESB. Tämä viitekehys on laajennettu yritys eli *Extended Enterprise*. Laajennettu yritys voisi olla esimerkiksi varaosia toimittava konserni, joka osana myyntiään tarjoaa SOA-perustaisen ohjelmistoratkaisun toimittajille ja loppupään myyjille, jossa sovelletaan tuotteiden hankintaketjuja.

Sovelluksia voidaan liittää BPEL-prosessiin soveltuvan protokollan kuten SOAP:n avulla kokonaan uudeksi BPEL-prosessiksi tai osaksi BPEL-prosessia käyttäen sopivia ohjelmointityökaluja. Tarvittaessa BPEL-prosessi tai yhteensopiva sovellus voidaan liittää osaksi ”yrityspalveluväylään” *Enterprise Service Bus* kokonaisvaltaisen yritysratkaisun luomiseksi.

BPEL syntyi aikaisemman kielen BPML:n (Business Process Modeling Language) vanavedessä täyttämään IBM:n ja Microsoftin tarpeita sopivien standardien saamiseksi tukemaan näiden toimijoiden olemassaolevia työnkulku ja integraatiomoottoreita (Wikipedia, 2008b). BPML:n integrointi IBM:n ja Microsoftin olemassaoleviin tuotteisiin ei ollut mahdollista BPML-kielen semanttisen formaaliuden vuoksi (BPEL:iin upotettavissa oleva Java mahdollistaa puuttuvan semantiikan korvaamisen) (Wikipedia, 2008c).

Työnkulkupalveluohjelmistot ovat olleet 1980-luvun lopulta osa pankkien, vakuutuslaitosten ym. työkaluja, sisältäen ydinkonseptit kuten aktiviteetit, tehtävät, roolit, määräajat, eskalaation, työlistat ja tehtävien vaatimisen(*task claiming*) (Silver, 2006). Nämä samat käsitteet ovat olennainen osa BPEL:ä ja erityisesti sen *TaskManager*-palvelua.

Oraclen BPEL-prosessimanageri koostuu joukosta palveluja, joista käyttäjän ja BPEL-prossessin välisen yhteyden muodostamisen väline on työlistapalvelu (*Worklist Service*). Työlistapalvelun rajapinnan avulla voidaan joko käyttää Oraclen valmista työlistasovellusta tai luoda sopiva web-käyttöliittymä JSP- ja servlet-sivuna.

Työlistapalvelu kuuluu osana työnkulkupalveluihin (Workflow Services), joihin sen lisäksi kuuluvat tehtävänhoito-, reititys-, identiteetti- ja ilmoituspalvelu. Yhteys BPEL-prosesseja ylläpitävään BPEL-moottoriin muodostuu moottorin TaskActionHandler-prosessista TaskManager- ja TaskRouting-palveluihin. Työnkulkupalvelut mahdollis-

tavat systeemien ja palvelujen integroimisen ihmistyön kanssa yksittäiseen prosessivirtaan.

Tavanomainen alusta web-palvelujen kehittämiseen on Java-pohjaisten sovellus- ja web-palvelimien käyttö. gSOAP-ympäristön kaltainen ratkaisu tarjoaa perinteisemmän lähestymistavan palvelujen liittämiseen, etunaan kevyempi käyttöjärjestelmätasoinen sovelluskehys. Skeleton-tyyppistä lähdekoodin generointia on yleisesti käytetty hajautettujen järjestelmien protokollia käyttävien sovellusten rakentamiseen. Luonnollisena seurauksena tämäntyyppisestä liitynnästä on, ohjelmoinnin yksityiskohtien lisääntymisen kustannuksella, suurempi muokattavuus ja sovitettavuus.

gSOAP-palvelun liittäminen BPEL-prosessiin ei käy aivan vaivatta, koska esimerkiksi JDeveloperin oletuksena luomaa muuttujat sisältävää WSDL-tiedostoa joutuu muokkaamaan yhteensopivaksi gSOAP-palvelun kanssa. Palvelun toimivuuden testaamisessa auttaa JDeveloperin Java-pohjaisen stub/skeleton-työkalun käyttö, jolla gSOAP-palvelun WSDL-tiedoston perusteella luotavissa olevalla Java-”stub”-sovelluksella on mahdollista testata ulkoista gSOAP-palvelua.

Web-palvelu voidaan toteuttaa gSOAP:ssa (kuten myös muissa Java- ja C-ympäristöissä) myös erillään web-palvelimesta, ns. *stand-alone*-palveluna halutussa portissa, käyttäen sisäänrakennettuja HTTP-, TCP/IP-pinoja ja rajoittuen käyttämään SOAP RPC:tä. Tämä tapa luonnollisesti vähentää järjestelmän kuormaa, jos palveluja ei ole monta.

Mikäli ohjelmoijalla on mielenkiintoa ja intoa syventyä yksityiskohtiin web-palvelujen ohjelmoinnissa, gSOAP-tarjoaa tähän kehyksen. Perinnesovelluksella viitataan joskus C/C++-kielellä toteutettuihin ratkaisuihin, mikä korostaa Javan merkitystä web-palvelujen tavallisimpana toteutuskielenä. Mikäli halutaan suorituskykyisempi ja vähemmän resursseja, kuten alustan muistia kuluttava ratkaisu esimerkiksi mobiili- ja PDA-laitteissa, gSOAP:n kaltaisilla ratkaisuilla on sijansa web-palvelujen kehittämisessä.

Viitteet

Agrawal, A. A., Amend, B. M., Das, O. M., Ford, A. E. M., Keller, A. E. C., Kloppmann, I. M., König, I. D., Leymann, I. F., Müller, O. R., Pfau, I. G., Plösser, S. K., Rangaswamy, O. R., Rickayzen, S. A., Rowley, B. M., Schmidt, S. P., Trickovic, S. I., Yiu, O. A. and Zeller, A. M. (2007). *WS-BPEL Extensions for People (BPEL4People), Version 1.0*.
URL: *http://www128.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/*

Barreto, C., Bullard, V., Erl, T., Evdemon, J., Jordan, D., Kand, K., König, D., Moser, S., Stout, R., Ten-Hove, R., Trickovic, I., van der Rijn, D. and Yiu, A. (2007). *Web Services Business Process Execution Language Version 2.0, Primer*.
URL: *http://docs.oasisopen.org/wsbpel/2.0/Primer/wsbpelv2.0Primer.pdf*

Bellwood, T. (2002). *UDDI Version 2.04 API Specification UDDI Committee Specification, 19 July 2002*.

Bradshaw, D., Kennedy, M. and West, C. (2005a). *Oracle BPEL Process Manager Developer's Guide 10g Release 2 (10.1.2) Part No. B14448-01,Appendix B Workflow and Notification Reference*, Oracle.
URL: *http://download-west.oracle.com/otndocs/products/bpel/bpeldev.pdf*

Bradshaw, D., Kennedy, M. and West, C. (2005b). *Oracle BPEL Process Manager Developer's Guide 10g Release 2 (10.1.2) Part No. B14448-01,Part I: Introduction and Concepts*, Oracle.
URL: *http://download-west.oracle.com/otndocs/products/bpel/bpeldev.pdf*

Bradshaw, D., Kennedy, M. and West, C. (2005c). *Oracle BPEL Process Manager Developer's Guide 10g Release 2 (10.1.2) Part No. B14448-01,Part II: Reviewing Key BPEL Development Concepts and Code Samples*, Oracle.
URL: *http://download-west.oracle.com/otndocs/products/bpel/bpeldev.pdf*

Bradshaw, D., Kennedy, M. and West, C. (2005d). *Oracle BPEL Process Manager Developer's Guide 10g Release 2 (10.1.2) Part No. B14448-01,Part III: Oracle BPEL Process Manager*, Oracle.
URL: *http://download-west.oracle.com/otndocs/products/bpel/bpeldev.pdf*

Chappel, D. (2004). *Enterprise Service Bus*, O'Reilly.

Dinesh, V., Dave, R., Berry, D., Dixit-Hardikar, P., Parikh, D., Eidson, B., Jain, S., Joshi, M., Stiel, H. and Xu, E. (2006). *Oracle® Enterprise Service Bus Quick Start Guide 10g (10.1.3.1.0) B28212-01 September 2006*, Oracle.

Engelen, R. (2005). *XML serialization for C/C++ with the gSOAP toolkit*, Dr Dobb's Portal.
URL: *http://www.ddj.com/cpp/184401909*

Engelen, R. (2008). *gSOAP 2.7.10 User Guide*.
URL: *http://www.cs.fsu.edu/ engelen/soap.html*

High, R., Kinder, S. and Graham, S. (2005). *IBM's SOA Foundation An Architectural Introduction and Overview Version 1.0 November, 2005*, IBM.
URL: *http://www.ibm.com/developerworks/webservices/library/wssoawhitepaper/*

Hirvonen, M. (2007a). *Käyttäjän vuorovaikutus BPEL-prosessin kanssa*, Kandidaatintutkielma, Joensuu yliopisto, Tietojenkäsittelytiede.

Hirvonen, M. (2007b). *Systemoinnin menetelmien harjoitustyö: Oracle BPEL tutorials, Usertasks 110 ja 132*, Joensuu yliopisto, Tietojenkäsittelytiede.

Juric, M., Mathew, B. and Sarang, P. (2004). *Chapter 5 of "Business Process Execution Language for Web Services"*, Packt Publishing.
URL: *http://www.oracle.com/technology/books/pdfs/bpel_ch5.pdf*

Ryan, F. (2007). *Web Services Business Process Execution Language Technical Introduction*.
URL: *http://www.oasisopen.org/committees/download.php/23068/WSBPEL Technical Overview for Developers and Architects Part 1 (Frank Ryan).pdf*

Silver, B. (2006). *BPEL4People Revisited,February 14th, 2006*.
URL: *http://www.brsilver.com/wordpress/2006/02/14/bpel4peoplerevisited/*

W3C (2000). *http://www.w3.org/TR/2000/NOTE-SOAP-20000508*.
URL: *http://www.w3.org/TR/2000/NOTE-SOAP-20000508*

Webopedia (2004a). *EAI: A Word Definition From Webopedia Computer Dictionary*.
URL: *http://www.webopedia.com/TERM/E/EAI.html*

Webopedia (2004b). *ESB: A Word Definition From Webopedia Computer Dictionary*.
URL: *http://www.webopedia.com/TERM/E/ESB.html*

WfMC (1999). *Terminology&Glossary, Document Number WFMC-TC-1011*.
URL: *http://www.wfmc.org/standards/docs/TC1011-term_glossary_v3.pdf*

Wikipedia (2008a). *Enterprise Service Bus*, WIKIPEDIA The Free Encyclopedia.
URL: *http://en.wikipedia.org/wiki/Enterprise_service_bus*

Wikipedia (2008b). *http://en.wikipedia.org/wiki/BPEL*.
URL: *http://en.wikipedia.org/wiki/BPEL*

Wikipedia (2008c). *http://en.wikipedia.org/wiki/BPML*.
URL: *http://en.wikipedia.org/wiki/BPML*

Woolf, B. (2007). *ESB-oriented architecture: The wrong approach to adopting SOA*, IBM.
URL: *http://www.ibm.com/developerworks/webservices/library/wssoaesbarch/*

Hakemisto