LAPPEENRANNAN TEKNILLINEN YLIOPISTO
Teknistaloudellinen tiedekunta
Tietotekniikan osasto

Työnkulkukoneen soveltuvuus sähköisen huutokauppajärjestelmän toteutuksessa

Diplomityön aihe on hyväksytty 30.10.2007.
Työn tarkastajat ovat Professori Juha Puustjärvi ja DI Daniela Grudinschi.
Työn ohjaaja on Professori Juha Puustjärvi.

Lappeenrannassa 19.2.2008

Jouko Huusko
Huvilakatu 26 B 25
80200 Joensuu
044-2748678

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto

Jouko Huusko

Työnkulkukoneen soveltuvuus sähköisen huutokauppajärjestelmän toteutuksessa

Diplomityö

2008

66 sivua, 35 kuvaa, 3 taulukkoa ja 2 liitettä

Tarkastajat: Professori, FT Juha Puustjärvi
Tutkijakoulutettava, DI Daniela Grudinschi

Hakusanat: Sähköinen huutokauppa, liiketoimintaprosessi, web-palvelut, työnkulkukone, ActiveBPEL
Keywords: Electronic auction, business process, web services, workflow engine, ActiveBPEL

Sähköiset huutokaupat ovat virtuaalisia markkinapaikkoja, jotka sijaitsevat jossain päin internetiä. Sähköistä huutokauppaa käydään yritysten välillä (B2B), yritysten ja kuluttajien välillä (B2C) sekä kuluttajien kesken (C2C). Tässä työssä sähköisellä huutokaupalla tarkoitetaan ensin mainittua, yritysten keskinäistä kaupankäyntiä.

Työn tarkoituksena on tutkia työnkulkukoneen soveltuvuutta sähköisen huutokauppajärjestelmän moottorina. Työssä perehdytään avoimen lähdekoodin ActiveBPEL-koneeseen, ja tutkimus tapahtuu suunnittelemalla, mallintamalla ja testaamalla liiketoimintaprosessi, joka rekisteröi ostajan ja myyjän tiedot järjestelmään. Toteutettava prosessi on yksi osa sähköistä huutokauppaa, mutta saman periaatteen mukaisesti olisi mahdollista toteuttaa myös kokonainen huutokauppa.

Tässä työssä tarkastellaan sähköistä huutokauppaa, joka perustuu web-palveluihin, ja jolla on selvä koordinaattori. Koordinaattori ohjaa toisia mukana olevia web-palveluja ja niiden ajettavia operaatioita. Korkean tason mallit kuvataan BPMN-notaation avulla, itse prosessi toteutetaan BPEL-kielellä. Prosessin mallinnuksessa ja simuloinnissa käytetään apuna ActiveBPEL Designer -ohjelmaa.

Työn tavoitteena on paitsi toteuttaa osa huutokaupasta, myös antaa lukijalle käsitys siitä liiketoimintaympäristöstä, johon huutokauppa kuuluu, sekä valottaa huutokaupan taustalla olevia teknologioita. Erityisesti web-palvelut ja niihin liittyvät käsitteet tulevat lukijalle tutuiksi.

ABSTRACT

Lappeenranta University of Technology
Department of Information Technology

Jouko Huusko

Suitability of workflow engine to implementation of electronic auction system

Master's thesis

2008

66 pages, 35 figures, 3 tables and 2 appendices

Examiners: Professor, Ph.D. Juha Puustjärvi
In research training, M.Sc. Daniela Grudinschi

Keywords: Electronic auction, business process, web services, workflow engine, ActiveBPEL

Electronic auctions are virtual marketplaces that reside somewhere in the internet. There are three different types of electronic auctions: between companies (B2B), between companies and customers (B2C), and between customers (C2C). In this thesis the electronic auction refers to the first mentioned auction between companies (B2B).

The aim of this master's thesis is to investigate the suitability of workflow engine as a motor of an electronic auction system. This thesis studies the open source ActiveBPEL engine, and the research is carried out by designing, modelling and testing a business process that registers information about the buyer and the seller into the system. The registration process is a specific part of electronic auction, but in accordance with the same principle it would be possible to carry out complete electronic auction.

This thesis examines an electronic auction based on web services that has a clear coordinator. Coordinator controls the involved web services and coordinates the execution of different operations on the web services involved in the operation. High level models are presented by means of BPMN notation, and the process is carried out with Business Process Execution Language (BPEL). Modelling and simulation of the process is executed by ActiveBPEL Designer software.

In addition to implementing a part of an electronic auction system, the aim of this master's thesis is to give the reader a comprehension about the business environment which electronic auction belongs to. Furthermore, the aim is also to introduce web technologies that are associated with electronic auction. Especially web services and concepts related to web services become familiar to the reader.

SISÄLLYSLUETTELO

LYHENTEET

B2B	Business-to-Business
B2C	Business-to-Customer
BPD	Business Process Diagram
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
BPMI	Business Process Management Initiative
BPML	Business Process Modeling Language
BPMN	Business Process Modeling Notation
C2C	Customer-to-Customer
HTTP	Hypertext Transfer Protocol
OASIS	Organisation for the Advancement of Structured Information Standards
OMG	Object Management Group
RPC	Remote Procedure Call
SOA	Service-Oriented Architectures
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
W3C	World Wide Web Consortium
WAPI	Workflow Application Programming Interface
WfM	Workflow Management
WS-BPEL	Web Services Business Process Execution Language
WS-CDL	Web Services Choreography Description Language
WSCI	Web Services Choreography Interface
WSDL	Web Services Description Language
WSFL	Web Services Flow Language
WWW	World Wide Web
XML	eXtensible Markup Language

1 JOHDANTO

Sähköisen huutokaupan juuret ulottuvat 1980-luvulle, jolloin niitä käytiin uutisryhmien ja sähköpostin välityksellä. Nykyisin huutokauppajärjestelmät ovat monimutkaisia ratkaisuja, joiden avulla yritykset voivat hankkia tai myydä tuotteita tai palveluja keskenään. Sähköistä huutokauppaa voidaan käydä yritysten välillä (B2B), yritysten ja kuluttajien välillä (B2C) sekä kuluttajien kesken (C2C), mutta tässä työssä sähköisellä huutokaupalla tarkoitetaan nimenomaan yritysten välistä kaupankäyntiä.

Myyjä- ja ostajayritysten on mahdollista asioida keskenään jopa täysin automatisoidusti sähköisen huutokaupan välityksellä, kun siinä käytettävät liiketoimintamallit määritellään ja toteutetaan tarkasti ja yksiselitteisesti. Myyjät ja ostajat voidaan korvata tietokoneohjelmilla, jotka esimerkiksi asettavat automaattisesti tarjouksia ennalta määritettyjen sääntöjen mukaan. Tämän avulla yritykset voivat saavuttaa kustannussäästöjä sekä lyhentää kaupankäyntiin kuluvaa aikaa. (Puustjärvi, 2006)

Erilaisia sähköisiä huutokauppajärjestelmiä on lukuisia, samoin kuin niissä käytettäviä teknologioita ja arkkitehtuuriratkaisuja. Tässä työssä esitellään web-palveluihin perustuva huutokauppa, jonka avulla yritysten välinen huutokauppa voidaan automatisoida mahdollisimman pitkälle. Kyseinen toteutusmalli on vielä suhteellisen uusi, eikä siihen liittyvää kirjallisuutta ole juurikaan olemassa. Sille on kuitenkin odotettavissa valoisaa tulevaisuutta, sillä monet web-palveluihin liittyvät teknologiat on suunniteltu alustariippumattomiksi, eivätkä ne siten vaadi käyttäjiltä suuria käyttöönottoinvestointeja.

1.1 Tavoite ja rajaukset

Työn tarkoituksena on tutkia työnkulkukoneen soveltuvuutta sähköisen huutokauppajärjestelmän moottorina, jolloin huutokaupan toiminnalliset osat ovat käytännössä ulkoisia web-palveluja. Tavoitteena on toteuttaa yksi osa huutokaupasta, jonka myötä koko järjestelmä voitaisiin toteuttaa saman periaatteen mukaisesti.

Sähköinen huutokauppajärjestelmä käyttää hyväkseen lukuisten web-palvelujen toimintoja, joten olennaisen tärkeää on pystyä koordinoimaan web-palvelujen suoritusta prosessissa. Web-palvelujen ja niiden välisten interaktioiden yhdistämiseen on olemassa kaksi tapaa, orkestrointi ja koreografia. (Juric, 2005) Tässä työssä tarkastellaan huutokauppamuotoa, jossa on selvä koordinaattori, jolloin on kyse orkestroinnista. Orkestroinnin toteuttamiseen tarvitaan jokin yhteinen kieli, johon on olemassa useitakin vaihtoehtoja. Tässä työssä käytetään jo lähes standardiksi muodostunutta BPEL4WS-kieltä, lyhyemmin sanottuna BPEL-kieltä.

Työnkulkukoneita on kehitetty lukuisia, niin avoimeen lähdekoodiin perustuvia kuin kaupalliseen tarkoitukseen suunnattuja, pääasiassa suurten yritysten lanseeraamia koneita. Tässä työssä perehdytään ActiveBPEL-koneeseen, joka on verkosta vapaasti ladattavissa oleva ja aktiivisesti kehitettävä ohjelmisto. ActiveBPEL-koneen kehittäneen yrityksen tuotteisiin kuuluu myös ActiveBPEL Designer, jota käytetään työn mallintamiseen ja simulointiin. (Active Endpoints, 2007)

Kolmannen sukupolven WWW web-palveluineen on vasta tekemässä tuloaan laajemmassa mittakaavassa, eikä tämän työn ideologian mukainen huutokauppajärjestelmäkään ole vielä nähnyt päivänvaloa. Web-palvelut voidaan toteuttaa useilla eri ohjelmointikielillä, mutta tämän työn kannalta erillisen web-palvelun luominen ei ole välttämätöntä; ActiveBPEL Designerin simulointityökalu mahdollistaa prosessin testaamisen sellaisenaan. Sen sijaan työn kokeellisessa osassa suunnitellaan ja mallinnetaan eräs liiketoiminnan kannalta oleellinen prosessi, jossa ostajalta ja myyjältä kerätään tietoja ja heidät rekisteröidään järjestelmään.

Työn tavoitteena on myös antaa lukijalle käsitys siitä liiketoimintaympäristöstä, johon huutokauppa kuuluu, sekä valottaa huutokaupan taustalla olevia teknologioita. Erityisesti web-palvelut ja niihin liittyvät käsitteet tulevat lukijalle tutuiksi. Työn keskeinen ajatus voidaan tiivistää tutkimuskysymyksen muotoon: ”*Miten ActiveBPEL soveltuu sähköisen huutokaupan moottoriksi?*”

1.2 Työn rakenne

Työ koostuu kahdesta osasta, teoriasta ja kokeellisesta osuudesta. Luvussa kaksi *sähköinen liiketoiminta* lukijalle selvitetään sitä, miten sähköinen huutokauppa sijoittuu osaksi yrityksen liiketoimintaa ja hankintatointa. Kolmannessa luvussa käydään läpi liiketoimintaprosesseja, niiden mallintamista sekä työnkulkuja, jotka yhdessä luovat perusedellytykset ymmärrettävälle ja toimivalle mallille. Neljäs luku syventyy sähköisen huutokaupan tietotekniseen osa-alueeseen, jotta lukijalle syntyy käsitys käytettävistä teknologioista, mallinnuskielistä sekä erityisesti web-palveluista, joiden ympärille tämän työn sähköinen huutokauppa on kehitetty. Neljännessä luvussa esitellään myös liiketoimintaprosessien kuvauskieli BPEL, jota käytetään työn soveltavassa osuudessa huutokaupan toteuttamiseen. Teoriaosuudessa käsitellään vielä yleisesti työnkulkukoneita ja niiden osuutta työnkulunhallintajärjestelmään, sekä tutustutaan ActiveBPEL-koneeseen, jonka käyttökelposuutta sähköisen huutokaupan moottorina työssä tutkitaan.

Jälkimmäisessä osassa työtä sähköisestä huutokauppajärjestelmästä suunnitellaan ja toteutetaan yksi osa, missä ostajalta ja myyjältä kerätään tietoja ja heidät rekisteröidään järjestelmään. Rekisteröintiprosessin toteutuksessa käytetään apuna ActiveBPEL Designer -ohjelmaa, jolla prosessi voidaan sekä mallintaa että simuloida.

Toteutettavan prosessin suunnittelu, mallinnus ja testaus käydään läpi vaiheittain havainnollisten kuvien avustamana. Korkean tason mallit kuvataan BPMN-notaation avulla, itse prosessi toteutetaan BPEL-kielellä. Testaus suoritetaan ActiveBPEL Designeriin sisäänrakennetulla simulointityökalulla, joka sisältää myös virtuaalisen työnkulkukoneen. Lopuksi selvitetään eteen tulleita ongelmia, havaintoja ja johtopäätöksiä, sekä esitetään kirjoittajan oma näkemys web-palvelupohjaisten huutokauppojen tulevaisuudennäkymistä.

2 SÄHKÖINEN LIIKETOIMINTA

Kalakota ja Robinson määrittelevät sähköisen liiketoiminnan kokonaisvaltaiseksi strategiaksi, jonka avulla yrityksen vanhoja liiketoimintamalleja pyritään kehittämään ja uudelleen arvioimaan. Uusien teknologioiden ja menetelmien avulla vanhoja toimintamalleja kehitetään niin, että asiakkaiden saama lisäarvo ja oman toiminnan tehokkuus sekä kannattavuus kasvavat. (Kalakota & Robinson, 2004)

Sähköinen liiketoiminta on kaikkea avoimien tietoverkkojen välityksellä tapahtuvaa kilpailuedun saavuttamiseen tähtäävää toimintaa. Tätä ovat muun muassa informaation välittäminen koko arvoketjussa prosessien tehostamiseksi ja lisäarvopalvelujen tuottamiseksi sekä puhtaat hyödykkeiden vaihdantaan tähtäävät transaktiot. (Karjalainen, 2000)

Sähköiset liiketoimintamallit eivät merkittävästi eroa perinteisistä liiketoimintamalleista. Ainoa merkittävä ero on se, että sähköisten työkalujen avulla myös monimutkaiset ja muutoin hankalat yhteistyökumppanuudet tulevat mahdolliseksi, kun tieto kulkee vaivattomammin yritysten välillä. (Kalakota & Robinson, 2004)

Karkeasti sähköinen liiketoiminta voidaan jakaa kolmeen osaan: kuluttajakauppa (B2C), yritysten välinen kaupankäynti (B2B) ja sisäisten prosessien tehostaminen (Aalto & al., 2000). Tässä työssä keskitytään yritysten väliseen kaupankäyntiin.

2.1 Sähköinen hankintatoimi

Sähköinen hankintatoimi on sähköisen liiketoiminnan osa-alue, jonka avulla yritys voi saavuttaa kilpailuetua ja kustannussäästöjä. Jotta sähköisestä hankinnasta olisi todellista hyötyä yritykselle, täytyy sillä olla selkeät päämäärät, joihin kyseisen menetelmän avulla pyritään. Sähköinen hankinta tuo mukanaan suoraan mitattavissa olevia etuja (esimerkiksi kustannus- ja hankintasäästöt), epäsuoraan mitattavissa olevia etuja (kuten

tehostunut tilausten haku ja seuranta) sekä aineettomia etuja, jotka eivät ole mitattavissa. (Lysons & Farrington, 2006)

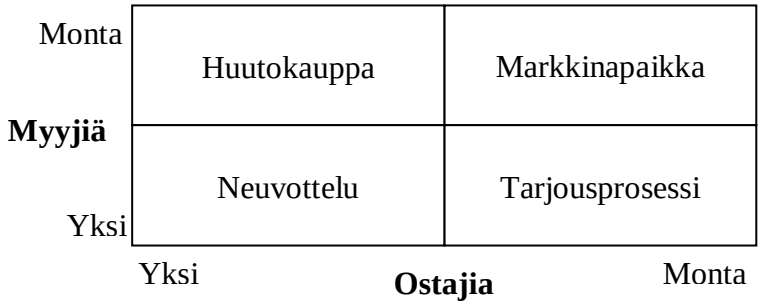
Sähköinen hankinta tarkoittaa tietoverkkojen hyödyntämistä tietyn tuotteen tai palvelun tilaamisessa, vastaanottamisessa, maksamisessa jne. Sähköistä hankintaa voidaan toteuttaa monin eri tavoin, mutta tärkeintä on, että järjestelmät voivat kommunikoida keskenään yli organisaatiorajojen. Rakenteellinen kuvauskieli XML on avoin standardi, jonka myötä organisaatiot voivat saumattomasti vaihtaa viestejä kauppakumppaniensa kanssa eri teknologioista riippumatta. (Lysons & Farrington, 2006)

Ennen sähköisen hankinnan strategian suunnittelua ja käyttöönottoa organisaatiolla on oltava koko hankintatoimea käsittävä hankintastrategia, jonka osaksi sähköinen hankinta voidaan suunnitella. Van der Merwen mukaan hankintastrategia on kilpailutilanneanalyysin pohjalta laadittava pitkän tähtäimen suunnitelma ja menettelyohje hankintojen toteuttamiseksi kilpailullisissa ja ympäristöllisissä olosuhteissa. Strategiaa muodostettaessa organisaatiossa tulee analysoida muun muassa se, mitkä tuotteet soveltuvat sähköisiltä kauppapaikoilta hankittaviksi, ja mitkä kannattaa hankkia muilla keinoin. (Van der Merwe, 2002)

2.2 Sähköinen huutokauppa osana hankintatointa

Yksi sähköisen hankinnan menetelmä on sähköinen huutokauppa. Se voi toimia perinteisen huutokaupan tapaan, jossa myyjät asettavat tuotteitaan myytäväksi ja ostajat voivat selailla niitä, tai päinvastoin, jossa ostajat tekevät tarjouspyyntöjä, ja myyjät vastaavat niihin kilpaillen toisia toimittajia vastaan reaaliaikaisesti. Jälkimmäistä kutsutaan käänteiseksi huutokaupaksi, ja se on yritysten välisissä huutokaupoissa yleisempi tapa. Sähköistä huutokauppaa voidaan soveltaa joko englantilaisen huutokaupan mukaisesti nousevin hintaportain, tai hollantilaisen huutokaupan tyyliin laskevin hintaportain määritellystä lähtöhinnasta hintaa pudottaen. (Lysons & Farrington, 2006)

Sähköisessä huutokaupassa voi olla osallistujia ympäri maailmaa, ja usein se suoritetaan rajatussa ajassa, esimerkiksi muutamassa tunnissa. Hinta markkinoilla on dynaamista, eli se määräytyy kysynnän ja tarjonnan mukaan. Kuvassa 1 on luokiteltu dynaamisen hinnoittelun eri muotoja, joita ovat markkinapaikka, huutokauppa, tarjousprosessi sekä kahdenvälinen neuvottelu. Sähköisten huutokauppojen huonona puolena tahtoo olla niiden keskittyminen hintaan hankinnan tärkeimpänä motiivina, minkä lisäksi lopullista kauppahintaa on usein hyvin hankala ennustaa etukäteen. (Karjalainen, 2000)



Kuva 1. Dynaamisen hinnoittelun tyypit. (Karjalainen, 2000)

2.3 Sähköisen huutokaupan erityispiirteet

Yksi merkittävimmistä eroista perinteisen ja sähköisen huutokaupan välillä on ostajan ja myyjän ”roolien vaihtuminen”, sillä yleensä sähköisessä huutokaupassa ostajat ilmoittavat vaatimuksensa haluamalleen tuotteelle, jonka jälkeen tavarantoimittajat tekevät tarjouksensa myymästään tuotteesta ostajan vaatimusten mukaisesti. Hinta ei välttämättä ole ainoa ratkaiseva tekijä, vaan ostajayritys voi ratkaista huutokaupan voittajan parhaaksi katsomillaan preferensseillä. Tällaista huutokauppamuotoa kutsutaan moniattribuuttiseksi huutokaupaksi. Hinnan lisäksi ostopäätökseen voivat vaikuttaa esimerkiksi tuotteen laatu, oheispalvelut, kuljetusehdot, toimitusaika, takuu, tai ylipäätään paras hinta-laatusuhde. (Kendall, 2001)

Uutta sähköisessä huutokaupassa on myös se, että siinä voi olla yhtä aikaa useita myyjiä ja ostajia, jolloin käynnissä on monia yksittäisiä transaktioita samanaikaisesti. Lisäksi tuotteita tai palveluja voi olla myytävänä kerralla useampia. Eräs tärkeä sähköisen

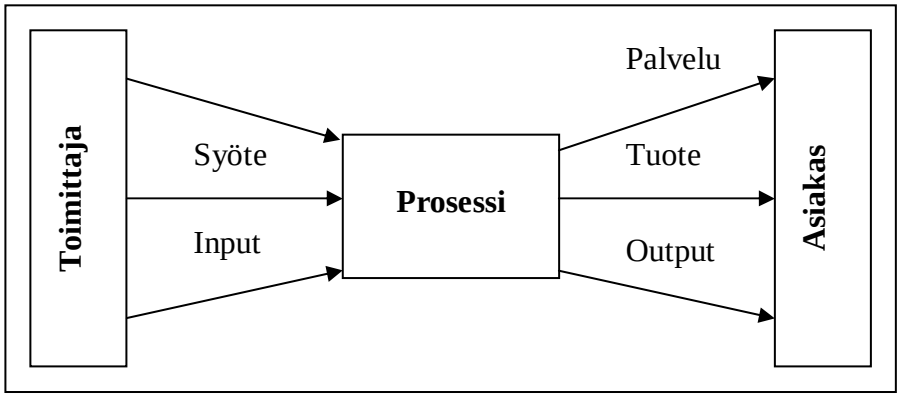
huutokaupan muoto on kombinatorinen huutokauppa, jossa myydään samanaikaisesti useita tuotteita siten, että huutajat voivat tehdä tarjouksia tuotteiden yhdistelmistä. Sähköiset huutokaupat mahdollistavat myös sellaisten tuotteiden ja ennen kaikkea palvelujen kauppaamisen, joita ei perinteisesti huutokaupoissa ole totuttu näkemään, kuten ohjelmointipalvelut. (Kendall, 2001)

Sähköisen huutokaupan merkittävimpiä hyötyjä liiketoiminnan kannalta on kannattavuuden lisääntyminen niin ostaja- kuin myyjäyritykselle alentuneiden transaktiokustannusten myötä. Lisäksi ostajayrityksen etsintä- ja yhteydenpitokustannukset sekä sopivien tavarantoimittajien etsintään kuluva aika ovat vähentyneet, ja toisaalta potentiaalisten tavarantoimittajien määrä on kasvanut, jolloin tarjolla on entistä enemmän ja kilpailukykyisempiä tuotteita. Käänteisen huutokaupan ansiosta ostajan on myös mahdollista reagoida nopeammin markkinoiden heilahteluihin. (Kendall, 2001)

3 LIIKETOIMINTAPROSESSIT

Liiketoimintaprosessi on joukko toisiinsa loogisesti liittyviä toimintoja ja niiden toteuttamiseen tarvittavia resursseja, joita tehdään jonkin liiketoiminnan kannalta tärkeän tuotoksen aikaansaamiseksi. (Laamanen, 2003) Liiketoimintaprosessista käytetään jatkossa myös lyhyttä muotoa prosessi. Prosessin ja projektin ero on siinä, että projekti on yksittäinen tapaus, kun taas prosessi käsittelee useita samankaltaisia tapauksia. (Savolainen & al., 1997)

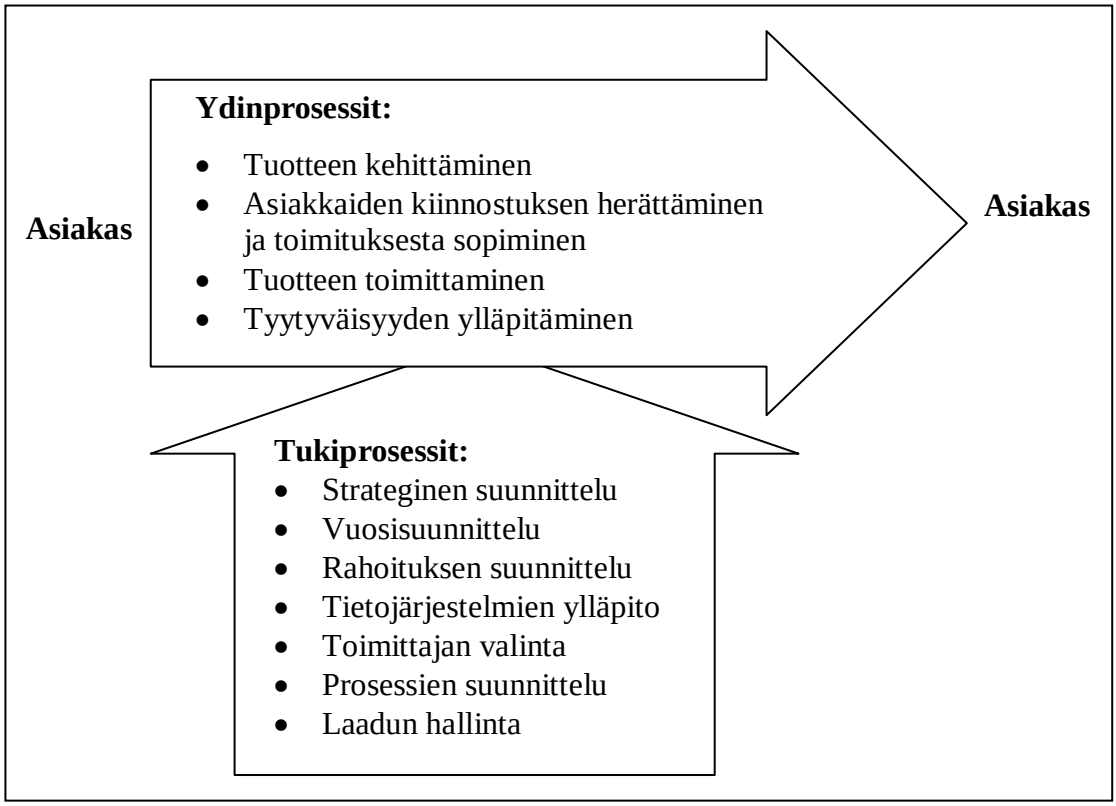
Prosessiajattelussa lähdetään liikkeelle asiakkaasta ja hänen tarpeistaan kuvan 2 mukaisesti. Mietitään, millaisilla tuotteilla ja palveluilla (output) ne voidaan tyydyttää. Suunnitellaan prosessi (toimenpiteet ja resurssit), joilla saadaan aikaan halutut tuotteet ja palvelut. Selvitetään, mitä syötteitä (input, tietoja ja materiaalia) tarvitaan prosessin toteuttamiseen ja mistä ne hankitaan (toimittajat). (Laamanen, 2003)



Kuva 2. Prosessi on sarja toimenpiteitä. (Laamanen, 2003)

Prosesseilla on joitakin tyypillisiä ominaispiirteitä. Ensinnäkin prosesseilla on määritetyt liiketoiminnalliset tuotosvaatimukset ja näillä tuotoksilla vastaanottajia eli asiakkaita. Asiakkaita voivat olla sisäiset tai ulkoiset asiakkaat, tai toiset prosessit. Toiseksi, vaikka prosesseja suoritetaan organisaation eri yksiköiden välillä, ne voivat ylittää organisaationaliset rajat. Kolmas tärkeä prosessin ominaisuus on sen mitattavuus; käytettyjä mittareita ovat muun muassa suorituskustannukset, läpimenoaika, laatu sekä tehokkuus. Lisäksi prosessilla on aina omistaja. (Laamanen, 2003)

Ulkoiseen asiakkaaseen suoraan liittyviä prosesseja sanotaan ydinprosesseiksi. Ydinprosessit voivat niiden laajuuden ja monimutkaisuuden takia koostua pienemmistä osaprosesseista, joita kutsutaan myös aliprosesseiksi. Aliprosessit puolestaan voivat koostua useammasta aktiviteetista, jotka vielä jakautuvat joukoksi pienempiä tehtäviä. Tehtävä on yhden ihmisen suoritettavissa oleva prosessin osa. Avainprosesseja ovat ne prosessit, joiden hyvä toimivuus on yrityksen menestykselle elintärkeitä. Tukiprosessit ovat nimensä mukaisesti ydinprosesseja tukevia prosesseja, jotka mahdollistavat ydinprosessien toiminnan. Kuvassa 3 on esitetty muutamia esimerkkejä organisaation ydin- ja tukiprosesseista. (Savolainen & al., 1997)



Kuva 3. Tukiprosessit luovat edellytykset ulkoista asiakasta palveleville ydinprosesseille. (Laamanen, 2003)

3.1 Liiketoimintaprosessien mallintaminen

Liiketoimintaprosessien mallintamisella tarkoitetaan yrityksen aktiviteettien ja niitä tukevan informaation tarkastelua tietyllä aikavälillä. Usein mallintamiseen törmätään

siinä vaiheessa, kun yrityksessä tulee tarve uudistaa vanhoja toimintatapoja, tehostaa prosesseja tai koordinoida teknologiaa sekä henkilöstöä uudelleen. Tällöin voidaan joko parantaa jo olemassa olevia prosesseja tai suunnitella ne kokonaan uusiksi. Mallintaminen on keino kuvata organisaation toimintaa, jotta sitä voidaan ymmärtää, analysoida ja kehittää. Mallintaminen ei kuitenkaan saa itsessään olla tavoite, vaan se on viestinnän väline. (Savolainen & al., 1997)

Hyvä prosessimalli sisältää prosessin kannalta kriittiset asiat, mutta ei mitään ylimääräistä. Vaikka kuva kertoo enemmän kuin tuhat sanaa myös mallinnuksessa, on silti joidenkin asioiden esittämiseen teksti paras vaihtoehto. Hyvä malli on myös tarkka, yhtenäinen (ei ristiriitoja), looginen sekä helposti muuteltavissa ja ymmärrettävissä. Mallista käy ilmi asioiden väliset riippuvuudet, se auttaa ymmärtämään sekä kokonaisuutta että omaa roolia tavoitteiden saavuttamisessa, ja lisäksi se edistää prosessissa toimivien ihmisten yhteistyötä. Prosessin kuvaaminen vaatiikin yhteistyötä niin liikkeen johdon kuin teknisten henkilöiden kesken, mikä on usein suurimpia haasteita koko prosessin toimimisen kannalta. (Savolainen & al., 1997)

Usein prosessien mallinnuksessa käytetään jotain visuaalista menetelmää, jolloin monimutkaisten yhteyksien esittäminen helpottaa käytännön työtä. Prosessien kuvaamiseen suositellaan joko UML-aktiviteettidiagrammia tai BPMN-notaatiota. Tässä työssä käytetään jälkimmäistä, joka esitellään tarkemmin luvussa 4.2.

Prosesseja voidaan mallintaa monella eri tasolla. Makrotason mallit kuvaavat korkealla abstraktiotasolla keskeisimpiä tehtäväkokonaisuuksia yksityiskohtien jäädessä piiloon. Alemmalle tasolle siirryttäessä yksityiskohtien määrä kasvaa ja konkreettisuus lisääntyy, kunnes aivan alimmalla mikrotasolla prosessimallit käsittelevät yksittäisiä työtehtäviä. White määrittelee mallintamisen tasot kolmeen osaan:

- *Prosessikartat (Process Maps)* ovat yksinkertaisia vuokaavioita aktiviteeteista.
- *Prosessikuvaukset (Process Descriptions)* ovat vuokaavioita, jotka sisältävät lisäinformaatiota, mutta eivät kuitenkaan niin paljon, että prosessin todellinen suorituskyky pystyttäisiin määrittelemään.

- *Prosessimallit (Process Models)* ovat vuokaavioita, jotka sisältävät riittävästi lisäinformaatiota kyseisen prosessin konkreettisesti analysoimiseksi, simuloimiseksi tai toteuttamiseksi.

BPMN tukee kaikkia näitä kolmea tasoa. (White, 2006)

3.1.1 Mallintamisen tarkoitus

Miksi liiketoimintaprosesseja sitten kannattaa mallintaa? Mallien avulla suurten kokonaisuuksien hahmottaminen on helpompaa, ja eri abstraktiotasojen avulla toimintoja voidaan tarkastella yksittäisestä tehtävästä kokonaisen organisaation toimintatapoihin asti. Mallintaminen auttaa näkemään missä järjestelmien tukea tarvitaan, mikä muuttuu ja mitkä järjestelmät muuttuvat. Se auttaa myös ymmärtämään liiketoiminnassa käytettävää tietoa, mistä se tulee, miten sitä käytetään ja miten se vaikuttaa liiketoimintaan (Vuolajärvi, 2006).

Mallintaminen parantaa myös yhteisymmärrystä ja kommunikaatiota. Standardeja mallinnusmenetelmiä käyttämällä voidaan välttyä kommunikaatio-ongelmilta, kun asioista puhutaan yhteisellä kielellä niin organisaation eri osastojen kuin eri organisaatioiden välilläkin. Mallinnuksessa on tiedon erilaisilla esitysmuodoilla, erityisesti kuvilla, suuri merkitys. Kun tieto on havainnollisesti esillä, liiketoimintaprosessien kehittämisessä mallintaminen eri näkökulmista lisää innovaatiomahdollisuuksia. Mallintaminen on myös oiva tapa osallistuttaa työntekijät suunnitteluun ja testaukseen. (Savolainen & al., 1997)

Liiketoimintaprosesseja mallinnetaan muun muassa organisaation toiminnankuvausta, laatujärjestelmän kuvausta, sekä prosessien nykytilan analysointia ja kehitystarpeiden tunnistamista varten (White, 2006). Vaikka mallintamisen tarkoituksena on usein parantaa nykyisiä prosesseja, jolloin nykymallista kehittyy kypsyttelyn kautta tavoitemalli, nykymallin merkitystä ei kannata väheksyä. Sen avulla opitaan kuvaamaan todellisuutta mallin avulla, mikä on tärkeää myös tulevaa tilaa mallinnettaessa. Lisäksi nykytilan kuvaaminen ongelmineen on arvokas tiedonkeruuvaihe, jolloin voidaan

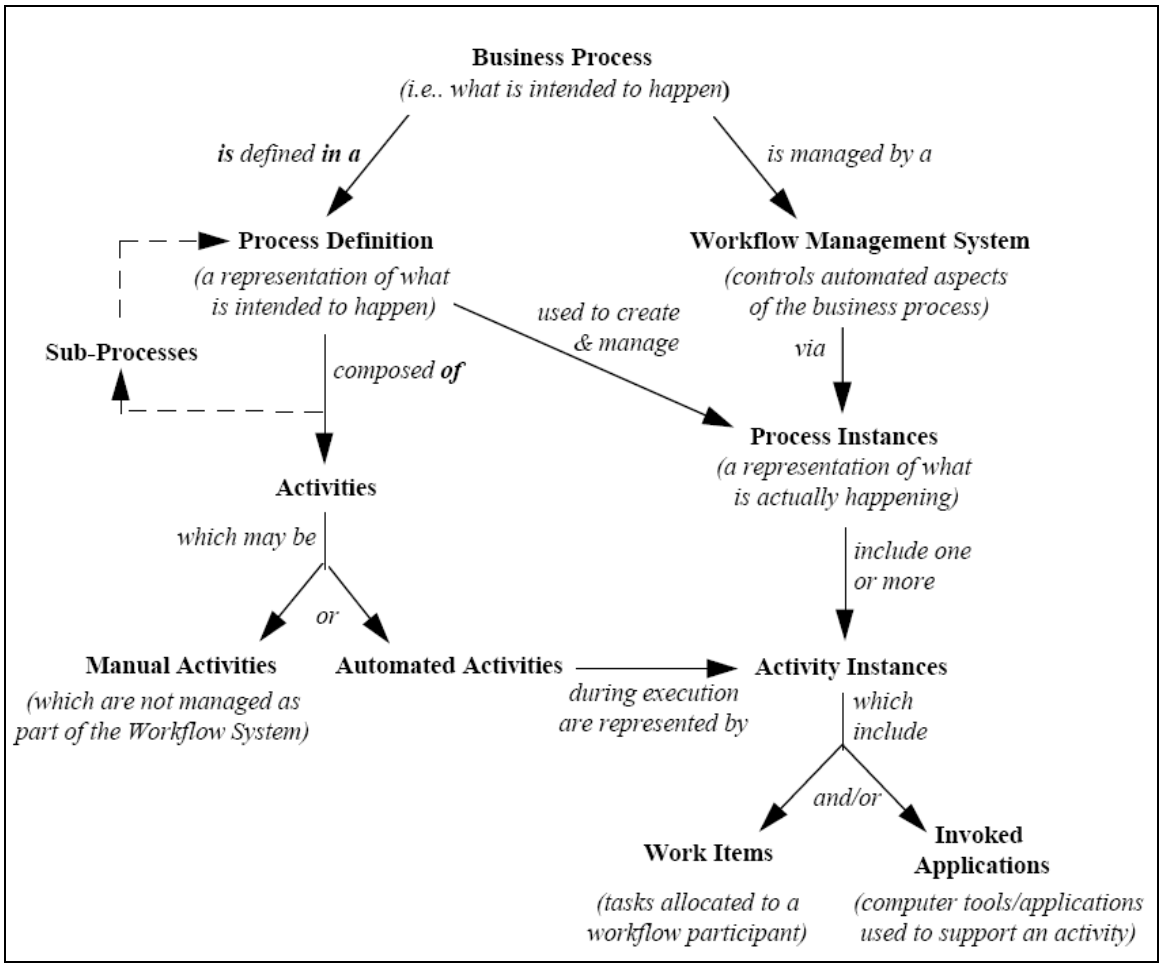
tunnistaa esimerkiksi lisäarvoa tuottamattomat (non-value added) aktiviteetit. (Savolainen & al., 1997)

3.1.2 Työnkulut

Työnkulut (workflow) ovat olennainen osa prosessien mallintamista. Työnkululla tarkoitetaan liiketoimintaprosessin tai sen osan automatisointia, jonka aikana dokumentit, tieto tai tehtävät välittyvät osallistujien välillä toiminnallisten sääntöjen mukaan. Osallistujat voivat tällöin olla joko ihmisiä tai koneita. Työnkulku liittyy aktiviteetti- ja tehtävätason mallintamiseen. (Hollingsworth, 1995)

Liiketoimintaprosessit ja työnkulut ilmentävät eri abstraktiotasoja organisaatiossa; liiketoimintaprosessit ovat hieman korkeammalla käsitetasolla työnkulkujen painottuessa tietotekniikkaan. Hollingsworthin mukaan liiketoimintaprosessien johtamisella ja työnkulkujen johtamisella ei kuitenkaan ole mitään eroa. Työnkulkujen johtamiseen kuuluu prosessien mallintamisen lisäksi prosessien uudelleensuunnittelu sekä työnkulkujen käyttöönotto ja automatisointi. (Hollingsworth, 2004)

Työnkulun hallintajärjestelmäksi kutsutaan työnkulkua jollain tavalla tukevaa järjestelmää, joka määrittää, hallitsee ja suorittaa työnkulkuja ohjelmistojen avulla (WfMC, 1999). Työnkulun hallintajärjestelmiin tutustutaan lähemmin luvussa 5.1. Kuvassa 4 on havainnollistettu, kuinka liiketoimintaprosessit ja niitä tukevien työnkulun hallintajärjestelmien peruskäsitteet liittyvät toisiinsa.



Kuva 4. Perustermistö ja niiden suhtautuminen toisiinsa. (WfMC, 1999)

Liiketoimintaprosessit kuvataan prosessimäärityksiksi, eli mitä oletetaan tapahtuvan. Prosessi voi jakaantua useisiin aliprosesseihin, jotka kuvataan vastaavalla tavalla. Alimmalla prosessitasolla määritetään prosessin varsinaiset työtehtävät eli aktiviteetit. Osa aktiviteeteista voidaan hoitaa automaattisesti, mutta kaikkia ei voida asettaa järjestelmän hallittavaksi. Työnkulun hallintajärjestelmä vastaa liiketoimintaprosessien automatisoitavien osien suorituksesta. Järjestelmä pitää yllä tietoa käynnissä olevista prosesseista ja huolehtii tapahtumien jakamisesta työtehtäviksi. Tehtävän voi suorittaa ihminen tai ohjelmisto. (WfMC, 1999)

4 SÄHKÖISESSÄ HUUTOKAUPASSA KÄYTETTÄVIÄ TEKNOLOGIOITA

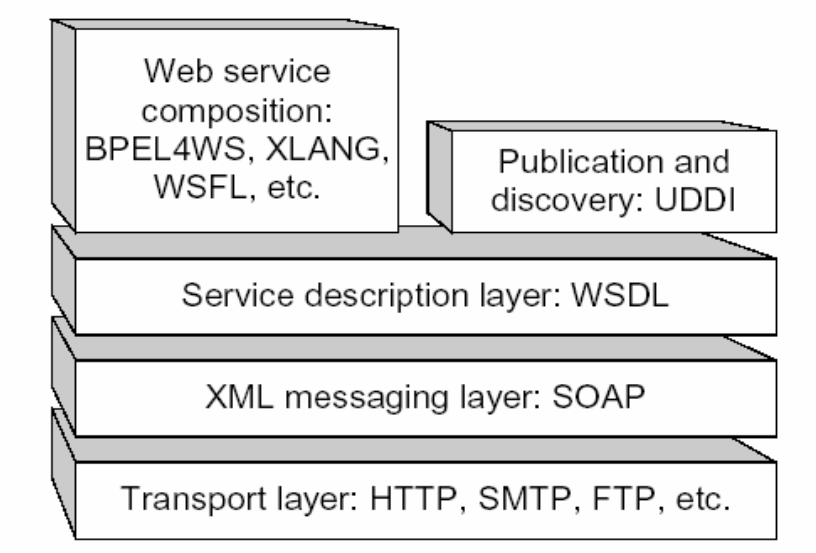
Web-palvelu (Web Service) tarkoittaa tietoverkkopohjaista palvelua, jolla on ohjelmallisesti käytettävä, standardi rajapinta (W3C, 2004b). Web-palvelut ovat hyvin keskeisessä osassa työn kokeellisessa osuudessa toteutettavassa sähköisen huutokauppajärjestelmän osassa. Tässä luvussa esitellään teknologioita, joiden avulla halutut liiketoimintaprosessit saadaan mallinnettua koneiden ymmärtämään muotoon siten, että ne niitä voidaan hallita automatisoidusti toisia web-palveluja hyödyntäen.

4.1 Web-palvelut

Web-palvelut ovat standardeja ja protokollia koneiden käytettävissä olevien palvelujen julkaisemiseen ja hyödyntämiseen webissä. Nykyisin webissä on pääasiassa sovelluksia, jotka palvelevat ihmistä, eli ne ovat ikään kuin käyttöliittymiä. Web-palvelut puolestaan ovat sovelluksia, jotka palvelevat muita sovelluksia verkossa, eli ne ovat myös koneita varten. Yritysten tietojärjestelmät on toteutettu monilla eri ohjelmointikielillä ja monille eri alustoille, jolloin niiden yhteensopivuus on vähintäänkin epävarmaa. Web-palvelujen avulla ulospäin näkyvä rajapinta voidaan määritellä standardoidusti koneiden välisen kommunikoinnin helpottamiseksi ja palvelujen löytämiseksi. Web-palvelu voi itsekin koostua useasta web-palvelusta, ja sen käyttö voi olla maksullista. Web-palvelujen yhdistämistä varten on kehitetty koreografia- ja orkestrointikieliä, esimerkiksi BPEL, joka esitellään myöhemmin tässä luvussa. (W3 Schools, 2007)

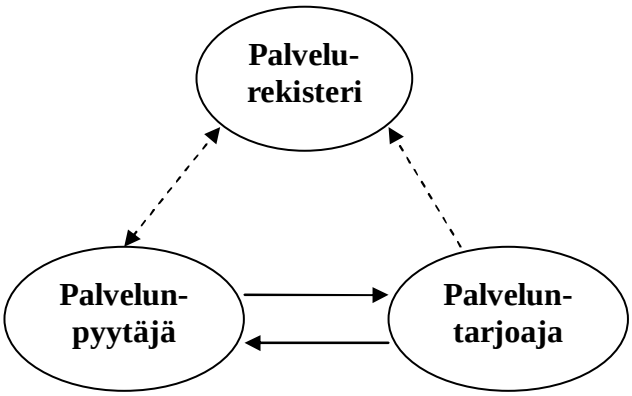
Web-palvelut eivät ole sinällään mikään mullistava keksintö, eivätkä ne korvaa olemassa olevia teknologioita, vaan tarjoavat uuden kerroksen näiden päälle. Web-palvelut perustuvat palveluorientoituneeseen arkkitehtuuriin (SOA), ja sen ydinteknologiat voidaan jakaa kolmeen luokkaan: viestintäprotokollaan, palvelun kuvauksiin ja palvelun löytämiseen, eli hieman yleistettynä saadaan yhtälö: web-palvelu = SOAP + WSDL + UDDI. Viimeksi mainittu mahdollistaa palvelujen julkaisemisen ja

löytämisen webistä, WSDL kuvaa palvelun rajapinnan ja SOAP:n avulla palvelun kanssa vaihdetaan tietoja XML-muotoisina viesteinä. (W3 Schools, 2007) Kuvassa 5 web-palvelut on esitetty pinorakenteena.



Kuva 5. Web-palvelujen teknologiapino. (Aalst, 2003)

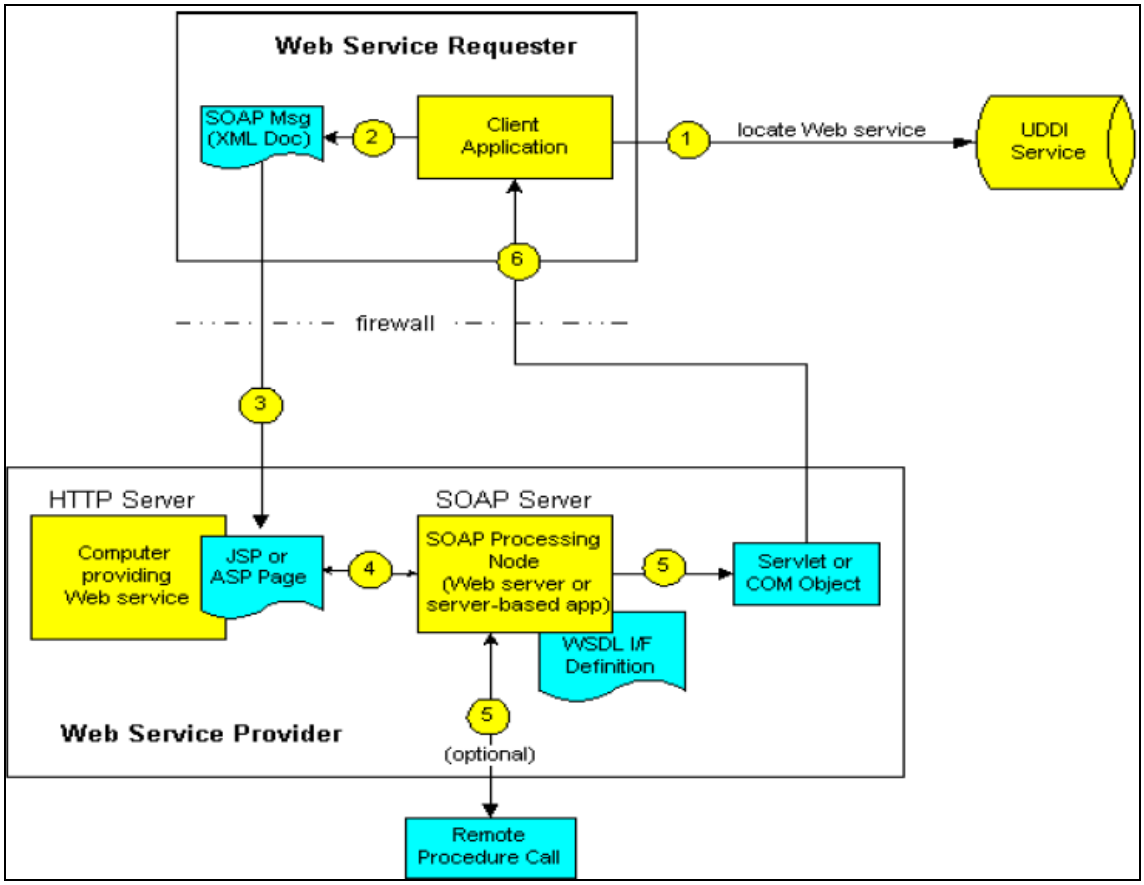
Web-palvelussa on kolme osapuolta: palvelurekisteri, palveluntarjoaja sekä palvelunpyytäjä, joka voi samalla olla myös palveluntarjoaja (kuva 6). Palvelurekisteri (UDDI) sisältää palvelun kuvaukset sekä määrittelee palvelun kuvauksen formaatin. Palveluntarjoaja kuvaa palvelut (WSDL), rekisteröi palvelut palvelurekisteriin sekä tarjoaa itse palvelut. Palvelunpyytäjä etsii palvelua rekisteristä, kutsuu sekä käyttää palvelua SOAP-viesteillä ohjeiden mukaisesti. Web-palvelujen infrastruktuurissa oletetaan, ettei sovelluksen tarvitse etukäteen tietää mitä palveluja lopulta käytetään. (W3C, 2004b)



Kuva 6. Web-palvelun osapuolet.

Web-palvelun toiminta voidaan esittää yksinkertaistettuna kuvan 7 osoittamalla tavalla:

1. Asiakas etsii UDDI-rekisteristä (jonne web-palveluntarjoaja kirjoittanut WSDL-kielellä kuvauksen palvelustaan ja rekisteröinyt sen) tarvitsemansa palvelun jollakin hakukriteerillä.
2. Asiakas muodostaa SOAP-viestin, jolla hän pyytää haluttua palvelua (jos palvelu on löytynyt, asiakkaalle syntyy palvelun WSDL-kuvauksen avulla SOAP-asiakasohjelma).
3. Asiakas lähettää SOAP-viestin palvelua tarjoavalle palvelimelle (asiakasohjelma kommunikoi palvelun tarjoajan SOAP-palveluohjelmiston kanssa).
4. Palvelin vastaanottaa viestin ja pyytää sopivaa ohjelmaa/oliota suorittamaan kutsutun palvelun vastaanottamillaan parametreilla.
5. Olio palauttaa tuloksen SOAP-palvelimelle, joka muodostaa vastauksesta SOAP-viestin ja lähettää sen palvelua pyytäneelle asiakkaalle.
6. Asiakas vastaanottaa vastausviestin, ja purkaa vastauksen. (Seppänen, 2002)



Kuva 7. Web-palvelun toiminta yksinkertaistettuna. (Seppänen, 2002)

4.1.1 SOAP

SOAP protokollan ensimmäinen versio (1.0) esiteltiin vuonna 1998, jonka kehittivät DevelopMentor, Userland ja Microsoft. Vuonna 2000 W3C julkaisi SOAP:sta version 1.1, ja saman vuoden lopulla W3C perusti työryhmän, joka tekisi SOAP:sta avoimen teknologian. (Saarela, 2002) Uusin versio SOAP:sta on 1.2, joka julkaistiin vuonna 2003.

SOAP tarjoaa rajapinnan sovellusten väliseen RPC-tyyliseen kommunikointiin. W3C:n määritelmän mukaan SOAP on ”kevyt protokolla, joka on tarkoitettu rakenteellisen tiedon siirtämiseen hajautetussa ympäristössä”. Se on yhteensopiva internetin standardien kanssa, joista erityisesti HTTP-protokolla on SOAP:n kannalta oleellinen. SOAP-sanomia voi kuljettaa muillakin protokollilla, mutta yleisimmin käytetään

HTTP:tä, sillä sen yleisyydestä johtuen palomuurit laskevat HTTP-liikenteen lävitseen, mikä vähentää ylimääräistä palomuurien säätämistä. SOAP on riippumaton asiakas- ja palvelinprosessien toteutustavasta (se on toteutettu yli 60 kielelle); riittää, että ne osaavat lähettää/vastaanottaa SOAP-sanomia. (W3C, 2004a)

SOAP-sanomia on kolmenlaisia. Asiakaskoneelta lähetetään palvelimen ohjelmalle kyselyviesti (SOAP request message). Jos palvelinkoneen ohjelma osaa käsitellä kyselyviestin, niin se lähettää vastaukseksi vastausviestin (SOAP response message). Jos palvelinkoneen ohjelma ei osaa käsitellä asiakaskoneelta saatua kyselyviestiä, niin palvelinkoneen ohjelma lähettää virheviestin (SOAP fault message). Myös SOAP-viestin käsittelijä palvelinkoneella voi lähettää virheviestin, jos esimerkiksi vastaanottava ohjelma ei ole toiminnassa. (W3C, 2004a)

4.1.2 WSDL

WSDL on XML-pohjainen kieli, jonka avulla web-palvelujen toiminta kuvataan. Uusin versio 2.0 hyväksyttiin kesäkuussa 2007 W3C:n viralliseksi suositukseksi. WSDL 1.1 on Microsoftin ja IBM:n W3C:lle tekemä standardiehdotus, joka on edelleen hyvin yleisesti käytössä; muun muassa web-palvelujen yhteentoimivuuskysymyksiin keskittyvä WS-I organisaatio käyttää sitä omissa profiilimäärityksissään (Basic Profile). (Systä, 2007b)

W3C:n määritelmän mukaan WSDL on ”XML-formaatti, jolla kuvataan web-palvelu päätepisteiden joukkona. Nämä operoivat viestien avulla, jotka sisältävät joko dokumentti- tai proseduurikeskeistä tietoa”. Käytännössä siis WSDL-dokumentti kuvaa muille sovelluksille web-palvelujen rajapinnan ja yhteystiedot. WSDL ei ota kantaa siihen, mitä viestit sisältävät eikä siihen, mitä protokollaa tiedonvälitykseen käytetään. Näin ollen WSDL ei ole sidottu esimerkiksi SOAP:iin tai HTTP:hen, joskin yleisimmin WSDL:ää käytetään juuri näitä protokollia käyttävien palveluiden kuvaamiseen. (W3C, 2001)

Web-palvelun WSDL-määritys koostuu seuraavista komponenteista:

- *Viesti (message)* on abstrakti kuvaus siirrettävästä tiedosta, joka kuvaa web-palvelun lähettämän tai vastaanottaman sanoman.
- *Porttityyppi (portType)* ryhmittelee web-palvelun sanomat loogisiksi operaatioiksi. Operaatio koostuu useista, toisiinsa liittyvistä sanomista. Porttityyppi kuvaa web-palvelun todellisen rajapinnan käyttäjään päin.
- *Sidos (binding)* määrittää sanomien siirrossa käytettävän tietoliikenneprotokollan ja tiedon formaatin.
- *Palvelu (service)* määrittää web-palveluun kuuluvat päätepisteet eli portit (port), joita voi olla yksi tai useampia. Portti tarkoittaa palveluportin verkko-osoitetta ja palvelun käyttämää tietoliikenneprotokollaa ja tiedon siirtoformaattia.
- *Tyypit (types)* määrittävät viestien käyttämät tietotyypit XML Schema -kielellä ilmaistuna. (W3C, 2001)

4.1.3 UDDI

Kun web-palvelu on luotu, puuttuu vain sitä käyttävät asiakkaat. UDDI määrittelee standardin tavan, jolla käyttäjät ja sovellukset voivat löytää palveluntarjoajia keskitetystä palvelurekisteristä. Määrittelyssä kerrotaan mitä tietoja palvelusta on kerrottava ja miten tieto esitetään. Määrittelyssä kuvataan myös rajapinta, jonka avulla palvelurekisterin tietoja voidaan hakea ja päivittää. Kuvauksen hakemiseen ja päivittämiseen käytetään standardia SOAP-rajapintaa. UDDI-määrittelyn takana olivat alun perin Microsoft, IBM ja Ariba, mutta nykyisin siitä vastaa kansainvälinen OASIS-organisaatio. (Saarela, 2002)

UDDI-palvelut voidaan jakaa kolmeen ryhmään:

1. *Valkoiset sivut* (white pages) sisältävät tietoa yrityksestä tai palveluntarjoajasta, kuten nimi, yhteystiedot, kuvaus, tunniste jne. (vrt. puhelinluettelo)
2. *Keltaiset sivut* (yellow pages) kertovat toimialaluokittelun, teollisuusalakoodin, tuotekoodin, maantieteellisen koodin jne. Sieltä voi siis hakea palvelun kuvauksen liikealan tai palvelun tyypin perusteella.

3. *Vihreät sivut* (green pages) sisältävät palvelun tekniset tiedot. (Saarela, 2002)

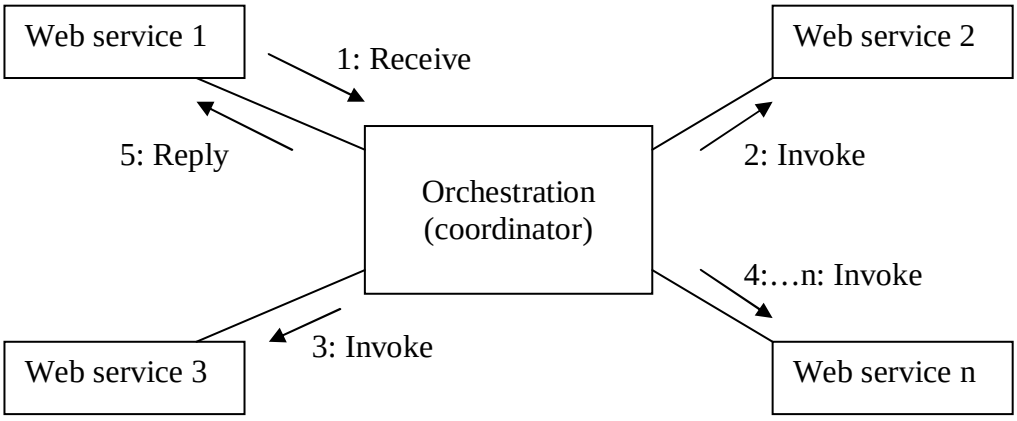
4.1.4 Orkestrointi vs. koreografia

Sähköisen kaupankäynnin nopea kasvu on pakottanut yritykset tarjoamaan palvelujaan sekä asiakkaille että yhteystyökumppaneilleen. Voidakseen reagoida nopeasti liiketoiminaan tarpeisiin, yritysten täytyy olla sähköisessä vuorovaikutuksessa ja integroida palvelujaan. Web-palvelujen yhdistäminen mahdollistaa korkeamman tason liiketoimintaprosessien luomisen käyttäen yksittäisiä web-palveluja rakennusosina. (Peltz, 2003) Web-palvelut yleensä ilmentävät tiettyjen sovellusten tai tietojärjestelmien operaatioita, joten web-palvelujen yhdistäminen tarkoittaa käytännössä kyseessä olevien yksittäisten web-palvelujen ja niiden toiminnallisuuksien integrointia (Juric, 2005).

Web-palvelujen ja niiden välisten interaktioiden yhdistämiseen on olemassa kaksi tapaa:

- *Orkestrointi* tarkoittaa suoritettavaa liiketoimintaprosessia, joka voi kommunikoida sisäisten ja ulkoisten web-palvelujen kanssa.
- *Koreografia* puolestaan on yhteistoimintaa, jossa jokainen osallistuja määrittää osuutensa toimintaan.

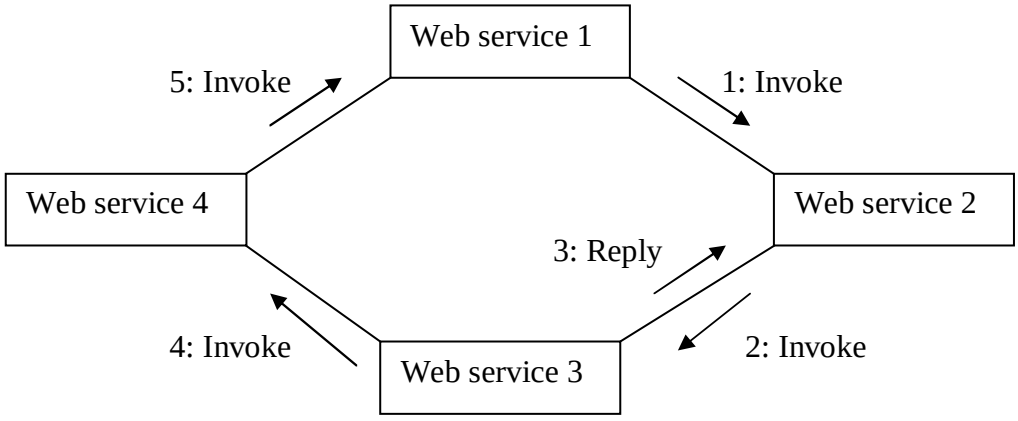
Orkestroinnissa keskusprosessi (joka voi olla myös yksi web-palvelu) toimii koordinaattorina, joka ohjaa toisia mukana olevia web-palveluja ja koordinoi niiden ajettavia operaatioita (kuva 8). Mukana olevat web-palvelut eivät tiedä mukanaolostaan prosessissa eivätkä siitä, että ne ovat osallisena korkeamman tason liiketoimintaprosessissa. Vain orkestroinnin keskuskoordinaattori on tietoinen prosessin tavoitteesta, joten orkestroinnissa on määritelty tarkoin operaatiot sekä mukaan kutsuttavien web-palvelujen järjestys. Itse prosessi voi sisältää interaktioita eri palvelujen - sisäisten tai ulkoisten - kesken, joten kyseessä voi olla pitkäkestoinen useita sovelluksia ja yrityksiä koskeva prosessi. (Juric, 2005)



Kuva 8. Web-palvelujen yhdistäminen orkestrointia käyttäen. (Juric, 2005)

Koreografiassa puolestaan joukko keskenään tasa-arvoisia palveluja muodostaa yhteyksiä, tekee mahdollisesti sopimuksia ja kommunikoi keskenään. Koreografiaa käytetään tyypillisesti julkiseen viestien vaihtoon, missä mukana voi olla useita web-palveluja ja osapuolia mukaan lukien asiakkaat, toimittajat tai yhteistyökumppanit. (Peltz, 2003)

Koreografiassa ei ole keskuskoordinaattoria, vaan jokainen mukana oleva web-palvelu tietää tarkalleen milloin sen on aika suorittaa omat operaationsa ja kenen kanssa se on vuorovaikutuksessa. Jokaisen koreografiaan osallistuvan täytyy olla tietoinen liiketoimintaprosessista, suoritettavista operaatioista, vaihdettavista viesteistä sekä viestien vaihdon ajoituksesta. (Juric, 2005) Koreografian toimintaa havainnollistetaan kuvassa 9.



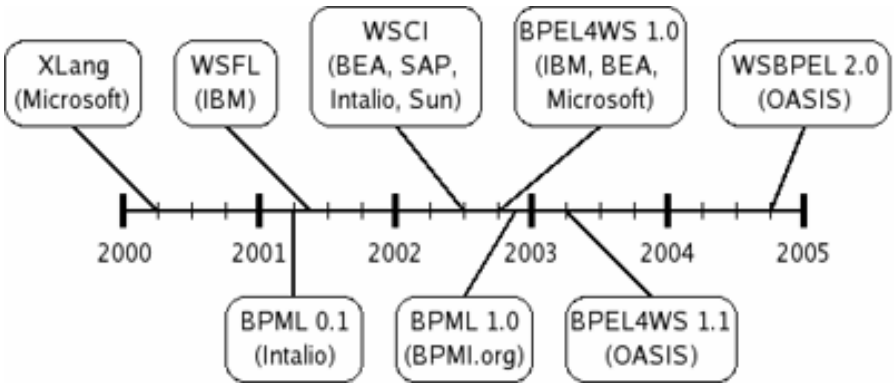
Kuva 9. Web-palvelujen yhdistäminen koreografiaa käyttäen. (Juric, 2005)

Liiketoimintaprosessien ajettavuuden kannalta orkestrointi on joustavampi web-palvelujen yhdistämistapa, ja tämän työn kokeellisessa osassa käytetään nimenomaan selkeän koordinaattorin sisältävää orkestrointia. Koreografiaan nähden orkestroinnilla on muun muassa seuraavia etuja:

- Osaprosessien koordinointia hallitsee tunnettu koordinaattori.
- Web-palveluja voidaan sisällyttää mukaan suurempaan liiketoimintaprosessiin ilman, että ne ovat tietoisia mukanaolostaan.
- Virhetilanteiden varalle voidaan syöttää mukaan vaihtoehtoisia skenaarioita. (Juric, 2005)

4.1.5 BPEL4WS

Käytettiinpä web-palvelujen yhdistämiseen sitten orkestrointia tai koreografiaa, tarvitaan sen toteuttamiseen jokin yhteinen kieli. Tähän tarkoitukseen on olemassa useita eri vaihtoehtoja, joista seuraavaksi esitellään työn kokeellisessa osassa käytettävä BPEL4WS, sekä verrataan sitä kahteen ”kilpailevaan” standardiin, BPML:ään ja WSCI:hin. Kuvassa 10 on esitetty viiden vuoden aikajanalla yleisimpiä standardeja, joilla on suurin piirtein samanlainen käyttötarkoitus.



Kuva 10. 2000-luvulla julkaistuja, orkestrointiin ja koreografiaan kehitettyjä standardeja. (Zwiers & de Vos, 2005)

BPEL4WS, lyhyemmin sanottuna BPEL, on kehitetty WSFL ja XLANG -kielten pohjalta ja näitä yhdistäen. WSFL on IBM:n Petri Net -malliin perustuva XML-

pohjainen formaatti web-palvelujen yhdistämisen kuvaamiseksi. XLANG puolestaan on Microsoftin Pi-Calculus -malliin pohjautuva WSDL-kielen laajennos, sisältäen elementin palvelun käyttäytymisen kuvaamiseksi osana liiketoimintaprosessia. (Systä, 2007a)

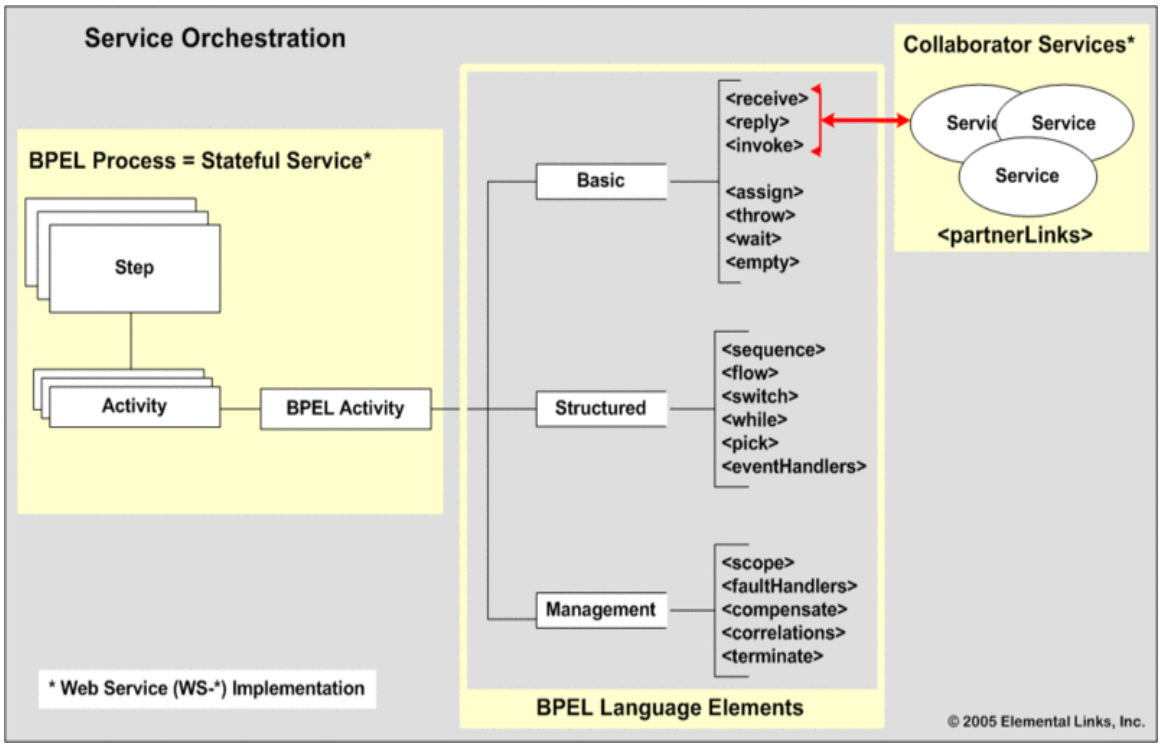
XML-pohjaisen BPEL-kielen kehitys alkoi IBM:n ja Microsoftin yhteistyöstä, jolloin syntyi BPEL4WS 1.0 web-palvelupohjaisten liiketoimintaprosessien kuvaamiseksi ja määrittämiseksi. Huhtikuussa 2003 BEA Systems, IBM, Microsoft, SAP ja Siebel Systems ehdottivat BPEL4WS 1.1 -kieltä OASIS-standardointiorganisaatiolle. Syyskuussa 2004 OASIS WS-BPEL tekninen komitea päätti nimetä spesifikaation uudelleen, jolloin nimeksi tuli WS-BPEL 2.0. Syynä tähän oli yhdenmukaistaa nimeämiskäytäntö muiden WS- alkuisten web-palvelustandardien kaltaiseksi. Toisaalta nimenvaihdos heijastelee huomattavia spesifikaatioon tehtyjä muutoksia, sillä BPEL4WS ei ole edes suoraan yhteensopiva version 2.0 kanssa. Huhtikuussa 2007 uusimmasta versiosta tuli virallinen OASIS-standardi. (Systä, 2007a)

BPEL:n avulla orkestrointia ja koreografiaa tukevat liiketoimintaprosessit voidaan kuvata kahdella eri tavalla:

- *Suoritettavat prosessit* mallintavat todellista vuorovaikutusta ja toimintaa osapuolten välillä. Ne noudattavat orkestrointiperiaatetta ja ne voidaan ajaa BPEL-moottorilla.
- *Abstraktit liiketoimintaprosessit* on tarkoitettu kuvaamaan prosessien välistä viestiliikennettä ja rajapintoja. Niitä ei voida ajaa, ja ne noudattavat koreografiaperiaatetta. (Juric, 2005)

BPEL-prosessin peruselementti on *aktiviteetti*. Kuvassa 11 erityyppiset BPEL-aktiviteetit on listattu kolmeen ryhmään, joita ovat perus-, rakenteiset ja hallinnointiaktiviteetit. Aktiviteettien lisäksi BPEL-prosessi voi sisältää muun muassa muuttujia ja tapahtuman käsittelijöitä. Ajettaessa erityisesti pitkäkestoista prosessia virhetilanteet eivät ole välttämättä harvinaisia, joten BPEL sisältää myös virheiden käsittelijöitä. Lisäksi jo suoritetut aktiviteetit saatetaan joutua peruuttamaan, koska ne ovat osa pidempää transaktiota, josta on jostain syystä jouduttu poistumaan. Ns.

kompensaation käsittelijät sallivat sellaisten tapahtumien määrittämisen, jotka tulee suorittaa tietyn tietoyksikön peruuttamiseksi ja datan palauttamiseksi sellaiseksi, mitä se oli ennen kyseisen työyksikön suorittamista. (Systä, 2007a)



Kuva 11. BPEL:n toiminta ja aktiviteetit. (Michelson, B. 2005)

BPEL-prosessikuvaus esittää prosessin yhden osapuolen näkökulmasta. Eri prosessien osapuolet (partners) esitetään WSDL-rajapintakuvauksena, joka kertoo tarjottavat palvelut ja miten palveluun saadaan yhteys. Näihin kuvauksiin viitataan <invoke> aktiviteetilla. BPEL-prosessi itsessään näkyy ulkopuolisille WSDL-rajapinnan kautta kutsuttavana sovelluspalveluna. BPEL-prosessi linkitetään loogisesti olemassa olevaan palveluun PartnerLink-elementin avulla. (Systä, 2007a)

BPEL on korkeantason ohjelmointikieli, jonka peruslähtökohtana on se, että itse web-palvelujen toteuttamisen sijaan ohjelmoijan pääfokus on olemassa olevien palvelujen orkestroinnissa (Systä, 2007a). Lisäksi BPEL, kuten muutkin nykyisistä SOA-tekniikoista, on alusta- ja kieliriippumaton, mikä mahdollistaa vuorovaikutuksen eri yritysten eri järjestelmien välillä. Näin esimerkiksi eri valmistajan CRM ja ERP -järjestelmät saadaan vaivattomasti keskustelemaan keskenään. (Seppänen, 2007)

BPEL:n lähin kilpaileva standardi on BPML. BPML on liiketoimintaprosessien mallintamisen metakieli, joka sisältää kaiken saman ja hieman enemmänkin kuin BPEL. BPML:n suurin ongelma on siinä, että sen uusin versio BPML 1.0 lanseerattiin kuukausi BPEL:in jälkeen. Kun Microsoftin, IBM:n ja SAP:n kaltaiset suuret organisaatiot olivat antaneet tukensa BPEL-kielelle, oli BPML:n kehittäneen BPMI-organisaation hyväksyttävä BPEL laajemmin tuettuna kielenä. (Zwiers & de Vos, 2005)

Kolmas standardi, joka on luotu suurin piirtein samaan tarkoitukseen kuin BPEL ja BPML, tunnetaan nimellä WSCI (lausutaan kuten ”whiskey”). WSCI on Sun Microsystems:n, SAP AG:n, BEA Systems:n ja Intalion yhdessä kehittämä kieli, jonka ensimmäinen versio julkaistiin heinäkuussa 2002. XML-syntaksiin perustuvan kielen avulla kuvataan useiden prosessiin osallistuvien osapuolien yhteinen koreografia määrittämällä web-palvelujen väliset viestit. WSCI-dokumentti määrittää web-palvelun ulkoisen rajapinnan käyttäjille ottamatta kantaa varsinaiseen prosessin suoritukseen. WSCI-koreografiaan voi kuulua joukko WSCI-dokumentteja, yksi jokaista osallistujaa kohti, kuten kuva 12 havainnollistaa. WSCI:llä ja BPEL:llä on monia samoja teknisiä vaatimuksia, kuten transaktioiden eheys, virhe- ja poikkeusten hallinta sekä tuki pitkäkestoisille prosesseille. (Peltz, 2003)

Kuva 12. WSCI-koreografiassa ei ole yhtä hallitsevaa osapuolta, vaan siihen kuuluu joukko WSCI-dokumentteja, yksi jokaista osallistujaa kohti. (Peltz, 2003)

Standardit kuten BPEL, WSCI ja BPML, on suunniteltu vähentämään web-palvelujen yhdistämisessä eteen tulevaa kompleksisuutta, ja sitä kautta lyhentämään tuotteiden tai palvelujen markkinoille saamiseen vaatimaa aikaa ja kustannuksia. Ne myös auttavat parantamaan liiketoimintaprosessien kokonaistehokkuutta ja -tarkkuutta. Ilman yhteisiä standardeja jokainen organisaatio joutuisi suunnittelemaan omat liiketoimintaprotokollansa, jolloin web-palvelujen yhdistäminen olisi joustamatonta ja työlästä. (Peltz, 2003)

Standardeja voidaan karkeasti ryhmitellä sen mukaan, kuvaavatko ne koreografiaa vai orkestrointia. BPEL sisältää molempia ominaisuuksia, mutta on selkeästi enemmän orkestrointisuuntautunut. WSCI puolestaan kallistuu enemmän koreografian suuntaan. Kuvassa 13 edellä esitellyt standardit on luokiteltu nelikenttään, missä yläpuoli painottuu koreografiaan ja alapuoli orkestrointiin. WSCI:tä ja BPML:ää voidaan käyttää samassa järjestelmässä, jolloin BPML määrittää liiketoimintaprosessit palveluiden takana, ja WSCI kuvaa palvelujen rajapinnan käyttömallin ulospäin. (Peltz, 2003)

Koreografia	**BPEL4WS** abstraktit prosessit	**WSCI**
Orkestrointi	**BPEL4WS** suoritettavat prosessit	**BPML**
	BPEL4WS	WSCI/BPML

Kuva 13. Standardien luokittelu sen mukaan, kuvaavatko ne koreografiaa vai orkestrointia.

4.2 BPMN

BPEL:n avulla määritellään työnkulkuja, mutta kuten ohjelmistokehityksessä yleensäkin, tulee suunnittelun edeltää varsinaista toteutusta, mikä tarkoittaa myös mallintamistarvetta. BPMI Notation Working Groupin kehittämä BPMN on liiketoiminnan mallintamiskieli, joka on jo melko laajasti käytössä. BPMN on spesifikaatio siitä miten liiketoimintaprosessin voi visualisoida, mutta se tarvitsee jonkun prosessinmäärityskielen alemmalle tasolle. Sen kehittämiseen osallistui 35 organisaatiota, ja ensimmäinen versio julkaistiin maaliskuussa 2004. Vuosien 2005 ja 2006 aikana BPMI:n ja OMG:n yhdistettyä voimansa BPMN:stä tuli OMG:n ylläpitämä, ja helmikuussa 2006 BPMN 1.0 muuttuikin OMG:n standardiksi. BPMN:n aktiviteettikaavio muistuttaa hyvin paljon UML:ää, mutta erona näillä on se, että BPMN on tarkoitettu nimenomaan liiketoimintaprosessien kuvaamiseen, kun taas UML:n aktiviteettikaavio on huomattavasti yleiskäyttöisempi. (Systä, 2007a)

Yksi BPMN:n kehittämisen päätavoitteista oli luoda silta liiketoimintaorientoituneen prosessimallintamisen ja IT-orientoituneiden ajettavien kielten välille siten, että se olisi helposti ymmärrettävissä myös tekniikkaan perehtymättömille käyttäjille ja johtoportaan työntekijöille. Toisaalta se tarjoaa keinon monimutkaistenkin prosessien mallintamiseen. Lisäksi BPMN-malli täytyy pystyä muuttamaan suoraviivaisesti esimerkiksi BPEL-määritykseksi, ja siihen tarkoitettua konversiotyökaluja on jo olemassakin. (Systä, 2007a)

BPMN määrittelee liiketoimintaprosessidiagrammin eli BPD:n, jonka avulla on mahdollista luoda graafisia malleja liiketoimintaoperaatioista. BPD sisältää seuraavanlaisia elementtejä:

1. Tapahtumaelementit (flow objects)
2. Tapahtumaelementtejä yhdistävät objektit (connecting objects)
3. Uimaradat (swimlanes)
4. Artifaktit (artifacts)

(White, 2004)

Tapahtumaelementit jakaantuvat tapahtumiin (event), aktiviteetteihin (activity) sekä vuon yhdistämiseen tai jakamiseen tarkoitettuihin päätössolmuihin (gateway). Tapahtumat kuvaavat yksinkertaisia tapahtumia, kun taas aktiviteetit kuvaavat tietyn osallistujan tekemän työn. Tapahtumia voi olla kolmenlaisia (kuvassa 14 vasemmalta oikealle lueteltuina): prosessin alkuvaihe, prosessin keskivaihe ja prosessin lopetusvaihe. (Systä, 2007a)

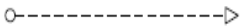
kuva 14. tapahtuma

kuva 15. aktiviteetti

kuva 16. päätössolmu

Tapahtumaelementtejä yhdistävät objektit kuvaavat itse vuota. Ne voivat olla aktiviteettien suoritusjärjestystä kuvaavia sekvenssivoita (sequence flow), kahden osallistuja välisiä viestivoita (message flow) tai dataa ja tekstiä yhdistäviä assosiaatioita (association). (Systä, 2007a)

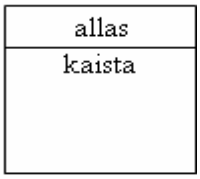
kuva 17. sekvenssivuo



kuva 18. viestivuo

kuva 19. assosiaatio

Uimarataa käytetään aktiviteettien organisointiin erilaisiin visuaalisiin osioihin. Nämä osiot eli altaat (pool) edustavat eri osallistujia kuvattavassa prosessissa. Tällaiset osallistujia kuvaavat osiot voidaan kuvan 20 mukaisesti edelleen jakaa eri osioihin eli kaistoihin (lane). (Systä, 2007a)



kuva 20. uimarata

Artifakteja ovat dataobjektit (data object), ryhmät (group) ja annotaatiot (annotation). Dataobjektit kertovat, miten aktiviteetit tuottavat tai vaativat tietoa. Ryhmiä käytetään dokumentaation kuvaamiseen, eivätkä sinällään vaikuta sekvenssivuohon. Annotaatioiden avulla puolestaan voidaan lisätä malliin tekstuaalista informaatiota diagrammin lukemista varten. (Systä, 2007a)

kuva 21. dataobjekti

kuva 22. ryhmä

kuva 23. annotaatio

5 TYÖNKULKUKONEET

Edellisessä luvussa esiteltiin web-palvelujen yhdistämiseen tarkoitettu orkestrointikieli BPEL, jonka avulla liiketoimintaprosessit voidaan kuvata suoritettaviksi prosesseiksi. BPEL-prosessien suorittamisesta vastaa työnkulkukone. Kirjallisuudessa BPEL-kieltä ymmärtävää työnkulkukonetta kutsutaan myös BPEL-moottoriksi sekä prosessimoottoriksi. BPEL-prosessi on XML-dokumentti, joka on suoritettavissa minkä tahansa valmistajan BPEL-moottorilla. (Systä, 2007a) Tässä työssä keskitytään pääasiassa työn kokeellisessa osassa käytettävään ActiveBPEL-koneeseen, joka esitellään luvussa 5.2.

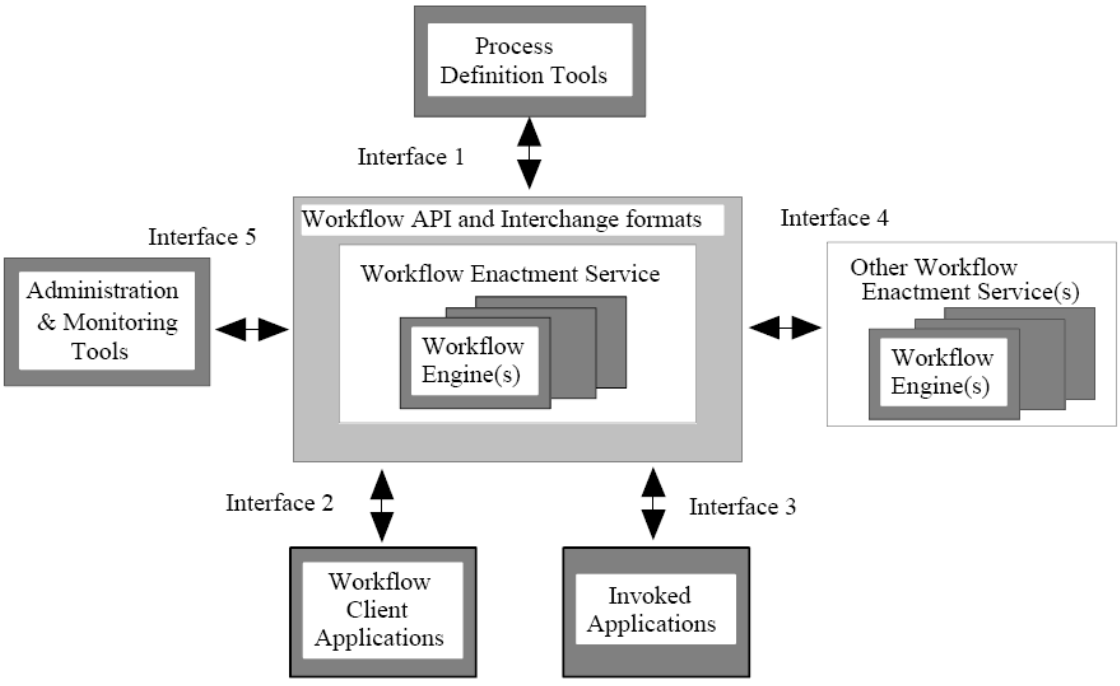
Työnkulkukone on ohjelmistopalvelu, joka tarjoaa ajonaikaisen ympäristön työnkulkuinstansseille, eli sen avulla voidaan hallita ja ajaa mallinnettuja liiketoimintaprosesseja. Työnkulkukone tulkitsee prosessimäärityksiä, luo ja valvoo prosesseja, vastaa tehtävien ajoittamisesta prosessissa sekä tarvittavien resurssien kutsumisesta mukaan ajettavaan prosessiin. Tehtävät voivat olla ihmisen tekemiä, puoliautomaattisia tai jonkin sovelluksen täysin automaattisesti suorittamia toimenpiteitä. Prosessia voi suorittaa useampikin työnkulkukone samanaikaisesti, jolloin ne jakavat prosessin tehtävät keskenään. (WfMC, 1999)

Erilaisia työnkulkukoneita on kehitetty lukuisia niin avoimeen lähdekoodiin perustuvia vapaasti saatavilla olevia koneita kuin kaupalliseen tarkoitukseen suunnattuja, pääasiassa suurten yritysten lanseeraamia koneita. Tunnettuja kaupallisia tuotteita ovat muun muassa Microsoftin BizTalk Server, Oraclen BPEL Process Manager ja Parasoftin BPEL Maestro, avoimen lähdekoodin työnkulkukoneita löytyy esimerkiksi java-source.net -sivustolta kymmenittäin, mukaan lukien tämän työn kannalta merkittävä ActiveBPEL.

5.1 Työnkulkukone osana työnkulun hallintajärjestelmää

Liiketoimintaprosessien yhteydessä mainittiin termi työnkulun hallintajärjestelmä, joka määrittää, hallitsee ja suorittaa työnkulkuja ohjelmistojen avulla. Koska työnkulkukone on osa laajempaa kokonaisuutta, on seuraavaksi syytä tutustua yleisellä tasolla työnkulun hallintajärjestelmään.

Yksittäinen liiketoimintaprosessi voi kestää sekunneista jopa useaan kuukauteen riippuen sen monimutkaisuudesta. Työkulun hallintajärjestelmä automatisoi liiketoimintaprosessia jakamalla sen tehtäviin ja kutsumalla ihmisiä tai koneita mukaan niiden käsittelyyn oikeassa järjestyksessä. (WfMC, 1999) Kuvassa 24 on eritelty tärkeimmät työnkulun hallintajärjestelmän osat Workflow Management Coalition julkaisemaan viitemalliin pohjautuen.



Kuva 24. Viitemalli työnkulun hallintajärjestelmille. (WfMC, 1999)

Viitemallin ydin on ajonaikainen työnkulun hallintapalvelu (Workflow Enactment Service), joka voi sisältää yhden tai useamman työnkulkukoneen. Muut komponentit ja sovellukset voivat hyödyntää hallintapalvelun tarjoamia palveluja työnkulun

ohjelmointirajapinnan kautta (WAPI). WAPI sisältää muun muassa API-kutsuja sovellusten ja työnkulkukoneen väliseen kommunikointiin sekä eri työnkulkukoneiden yhteistoimintaan vaadittavia protokollia. (WfMC, 1999)

Workflow Management Coalition viitemallin kuuluu lisäksi yleisiä komponentteja, joita työnkulun hallintajärjestelmät usein sisältävät. Tällaisia ovat prosessin määritystyökalu, työnkulun asiakasohjelmisto (Workflow Client Application), käynnistettävä ohjelmisto (Invoked Application), erilaiset hallinta- ja seurantatyökalut (Administration & Monitoring Tools) sekä ulkopuoliset työkulun hallintapalvelut (Other Workflow Enactment Services). Järjestelmään voi kuulua muitakin yrityskohtaisesti räätälöityjä komponentteja, tai se voi yksinkertaisimmillaan olla vain viestijärjestelmä, joka kontrolloi sanomien kulkua ja järjestystä. Yleisesti voidaan kuitenkin todeta, että työnkulun hallintajärjestelmän tulee hyödyttää sekä yritystä, asiakkaita, että sen käyttäjiä. (WfMC, 1999)

Työnkulkujärjestelmä voi olla myös hajautettu, jolloin järjestelmien tulee pystyä kommunikoimaan keskenään läpi prosessin. Hajautettuja järjestelmiä tarvitaan silloin, jos liiketoimintaprosessit koostuvat eri toimipaikkojen tai yritysten välisestä yhteistoiminnasta. Järjestelmien yhdistämiseen on olemassa useita eri tapoja, mutta tässä työssä hajautettuja järjestelmiä ei sen syvällisemmin käsitellä.

5.2 ActiveBPEL

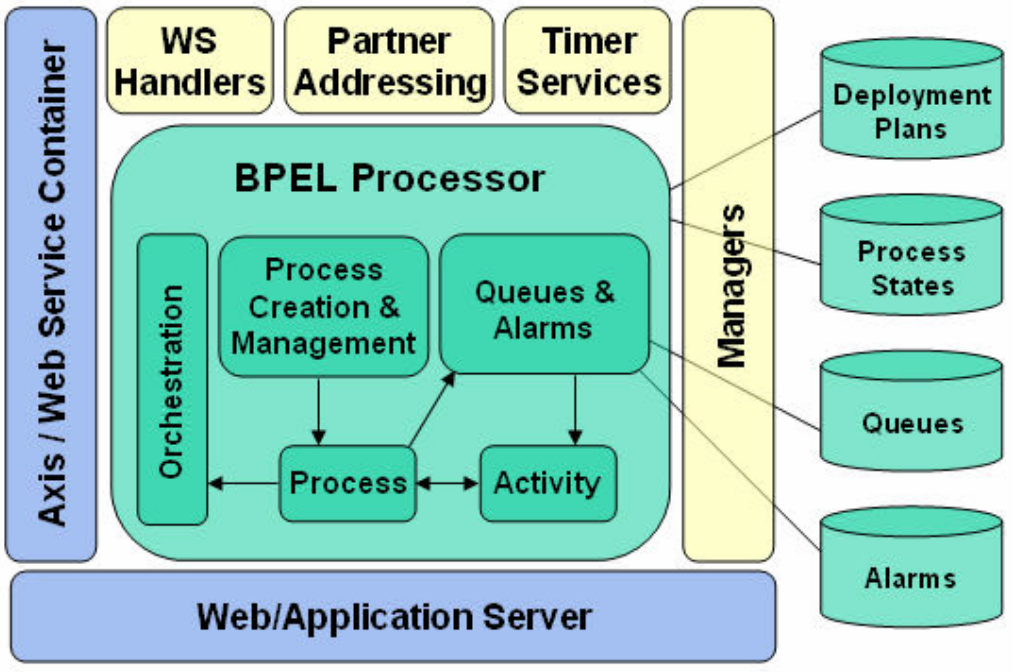
ActiveBPEL on avoimeen lähdekoodiin perustuva java-pohjainen BPEL-kone, jonka lisensioinnista ja jakelusta vastaa ActiveBPEL, LLC -organisaatio. ActiveBPEL-koneen teknologian on kehittänyt Active Endpoints, Inc. (AEI), jonka alaisuuteen myös ActiveBPEL, LLC -organisaatio kuuluu. ActiveBPEL-koneen uusin versio kirjoitushetkellä on 4.0, joka julkaistiin 5. heinäkuuta 2007. (Active Endpoints, 2007)

ActiveBPEL-kone tarjoaa ajonaikaisen ympäristön, minkä avulla BPEL-kielellä kirjoitettuja prosessikuvauksia voidaan lukea ja suorittaa. Kone on yhteensopiva sekä BPEL4WS 1.1 että WSBPEL 2.0 määritysten kanssa. ActiveBPEL vaatii toimiakseen

jonkin servlettejä käsittelevän servlet container:n (esimerkiksi Apache Tomcat) sekä kehitysympäristön (Java Development Kit). (Active Endpoints, 2007)

ActiveBPEL-koneen arkkitehtuuriin kuuluu kolme päätekijää: kone, prosessit ja aktiviteetit. Kone koordinoi yhden tai useamman BPEL-prosessin suorittamista. Prosessit puolestaan muodostuvat aktiviteeteista. Kone luo uuden prosessin BPEL-prosessikuvauksen (XML-tiedosto) perusteella ja sen jälkeen ajaa prosessin (kuten jo aiemmin todettiin, BPEL-prosessi voi olla myös abstrakti, mutta ActiveBPEL-kone suorittaa pelkästään ajettavia prosesseja). (Active Endpoints, 2007)

ActiveBPEL-kone koostuu komponenteista, jotka on eritelty kuvassa 25. Koneen ydin sisältää liiketoimintalogiikan ja tietämyksen BPEL-dokumentin käsittelemisestä. Oikealla olevat tietokantaelementit edustavat yleisiä pysyväismuisteja; ActiveBPEL-koneen arkkitehtuuri on lisäosilla laajennettavissa oleva, jolloin erilaiset pysyvyyden hallintaohjelmat voivat käyttää eri muistimekanismeja. Apache Axis toimii rajapintakomponenttina, sillä se osaa käsitellä järjestelmään tulevat SOAP-sanomat ja muodostaa järjestelmästä lähtevät SOAP-sanomat. (Active Endpoints, 2007)



Kuva 25. ActiveBPEL-koneen komponentit. (Active Endpoints, 2007)

ActiveBPEL-koneen toinen päätekijä, prosessi, koostuu seuraavista komponenteista:

- *Partner links* -elementti määrittelee web-palvelujen väliset yhteydet rajapintatasolla.
- *Partner*s -elementti määrittää prosessin osapuolet.
- *Muuttujat (variables)* sisältävät arvoja.
- *Korrelaatiot* (*correlation sets)* identifioivat liiketoimintaprosessit. Korrelaatiota käytetään, jotta eri viestit menisivät oikeille instansseille kuljetusprotokollasta riippumatta.
- *Virheiden käsittelijä (fault handler*) kertoo kuinka virhetilanteista selvitään.
- *Kompensaation käsittelijä (compensation handler*) mahdollistaa jo suoritettujen liiketoimintaprosessien peruuttamisen ja datan palauttamisen sellaiseksi, mitä se oli ennen suoritusta.
- *Tapahtuman käsittelijä (event handler*) käsittelee vastaanotettuja viestejä eli tapahtumia sekä hälytyksiä.
- *Ylimmän tason aktiviteetti (top-level activity)* on tyypillisesti rakenteinen aktiviteetti, joka sisältää muita aktiviteetteja, kuten perusaktiviteetteja, jotka kuvaavat varsinaisia operaatioita. (Active Endpoints, 2007)

6 SÄHKÖISEN HUUTOKAUPAN TOTEUTUS

Täysimittainen huutokauppajärjestelmä on valtava kokonaisuus; esimerkiksi eBay verkkohuutokauppadokumentaatio sisältää yli 1800 sivua. Tämän työn kokeellisessa osassa sähköisestä huutokauppajärjestelmästä toteutetaan pieni osa, missä ostajalta ja myyjältä kerätään tietoja ja heidät rekisteröidään järjestelmään.

Kokeellinen osuus rakentuu seitsemästä alaluvusta, joissa käydään muun muassa prosessin mallinnus, sen tuloksena syntyvä BPEL-kielinen prosessitiedosto, prosessin rajapintakuvaus sekä testaus vaiheittain läpi havainnollisten kuvien avustamana. Sitä ennen pohjustetaan käytettäviä toteutusmenetelmiä sekä esitellään palvelun kuvaus. Lopuksi arvioidaan toteutuksessa syntyneitä tuloksia, tuodaan esiin tutkimuksen aikana ilmenneitä ongelmia sekä pohditaan lopputulosta tutkimusongelmaan viitaten.

6.1 Toteutusmenetelmät

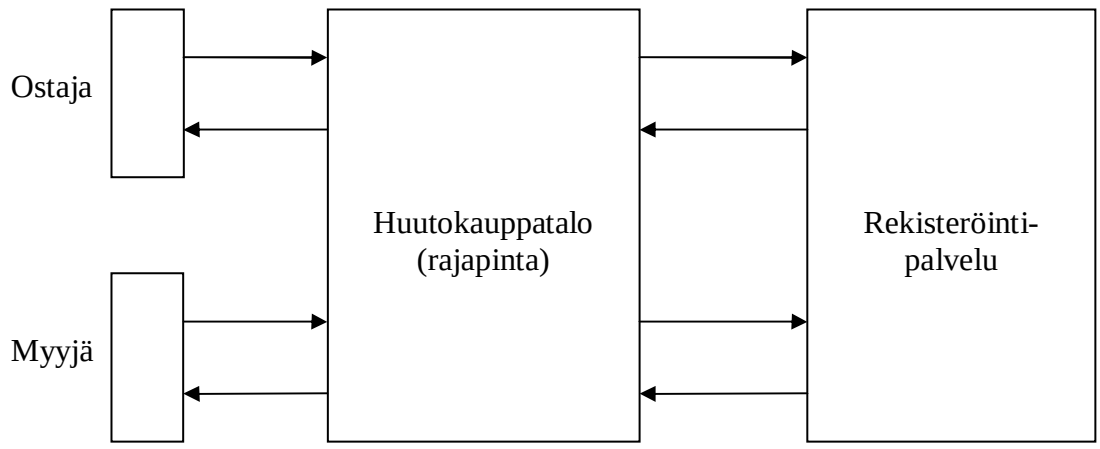
Työssä toteutettavan sähköisen huutokaupan osaprosessin, rekisteröintiprosessin suunniteluun, mallintamiseen ja testaukseen käytetään ActiveBPEL Designer -ohjelmaa, joka on myös ActiveBPEL-koneen kehittäneen Active Endpoints organisaation tuote. ActiveBPEL Designer on liiketoimintaprosessien mallinnustyökalu, joka muodostaa automaattisesti valmista BPEL-koodia käyttäjän piirtämien graafisten kuvioiden pohjalta. Ohjelma perustuu avoimen lähdekoodin Eclipse ohjelmointiympäristöön, ja se pitää sisällään selkeän käyttöliittymän sekä monipuolisen ohjeistuksen.

Työssä toteutettavan prosessin suunnittelussa ja mallinnuksessa on hyödynnetty OASIS-organisaation WS-BPEL 2.0 -standardin verkkosivuilta löytyvää *Auction service* esimerkin WSDL-dokumenttia. WSDL-dokumenttia tarvitaan palvelun rajapinnan kuvaamiseen, ja se on välttämätön työn toteutuksen kannalta. Esimerkin WSDL-dokumentti ei ole kuitenkaan täydellinen, vaan siitä puuttuu kokonaan types, service ja binding -komponentit. Valmis WSDL-dokumentti löytyy kokonaisuudessaan liitteestä 1.

Erillistä ostajan ja myyjän tiedot rekisteröivää web-palvelua ei tässä työssä konkreettisesti toteuteta, sillä ActiveBPEL Designer sisältää prosessin simulointityökalun, joka kuvastaa prosessin käyttäytymistä eri syötteillä aivan kuin sitä ajettaisiin asiakasohjelmalla. Web-palvelu voitaisiin toteuttaa useilla eri ohjelmointikielillä, mutta hyvin usein web-palvelut ohjelmoidaan Javalla. Koska ActiveBPEL Designer sisältää virtuaalisen työnkulkukoneen, ja se on saman yrityksen kehittämä tuote kuin ActiveBPEL-kone, voidaan tämän työn puitteissa olettaa, että mikäli prosessi toimii toivotulla tavalla simulaattorissa, se toimii myös ActiveBPEL-koneen pyörittämänä.

6.2 Palvelun kuvaus

Kuvassa 26 on esitelty rekisteröintiprosessin osapuolet liiketoimintaprosessidiagrammin eli BPD:n avulla. Ostaja ja myyjä kommunikoivat rekisteröintipalvelun kanssa huutokauppatalon kautta, joka toimii prosessissa koordinaattorina ja rajapintana välitettäville sanomille. Rekisteröintipalvelu on tässä erillinen web-palvelu, jonka huutokauppatalo kutsuu mukaan prosessiin tietyssä vaiheessa. Prosessissa ostajalta ja myyjältä, jotka ovat niin ikään web-palveluja (automatisoituja tai käyttäjän toimia vaativia), kerätään tietoja tiettyyn huutokauppaan liittyen, välitetään tiedot eteenpäin rekisteröintipalvelulle ja lopuksi lähetetään ilmoitus rekisteröinnistä ostajalle ja myyjälle.



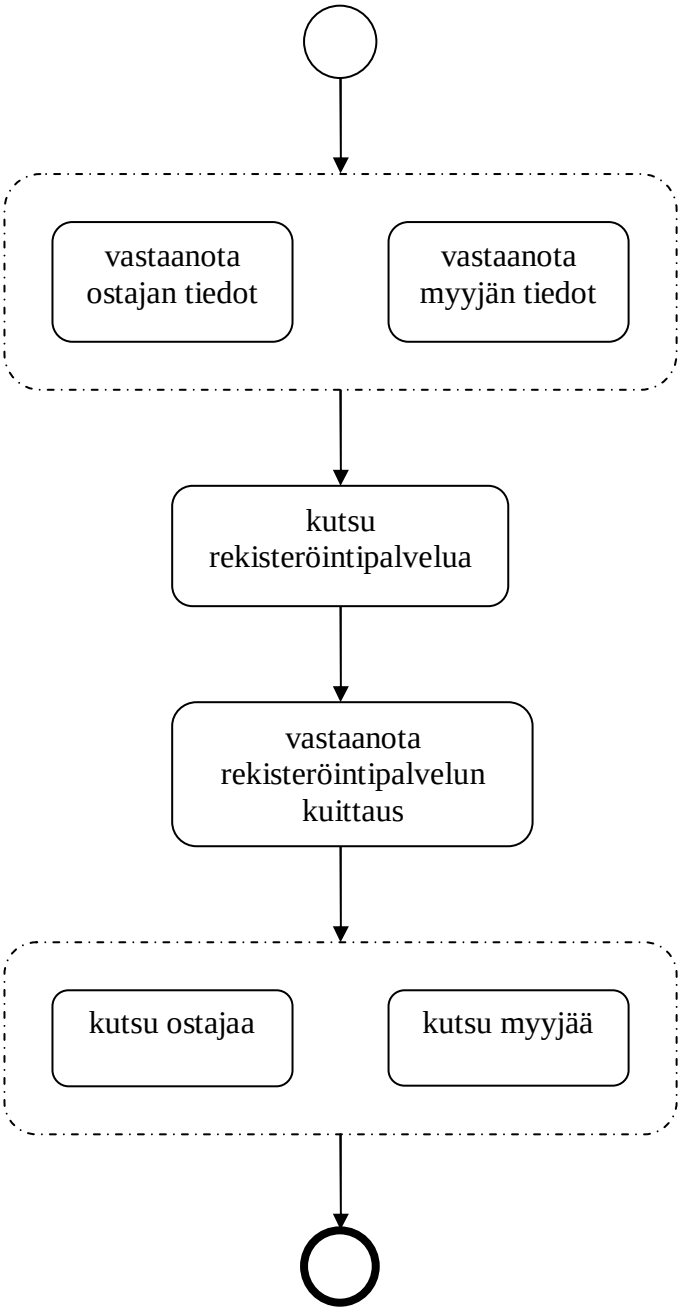
Kuva 26. Rekisteröintiprosessin osapuolet.

Prosessi voi käynnistyä vastaanottamalla joko ostajan tai myyjän tiedot, niiden järjestyksellä ei ole väliä. Jokainen huutokauppa sisältää yksilöllisen tunnisteen (ID), joten ostajan ja myyjän täytyy kertoa haluttu tunniste muun tiedon ohella. Kun sekä ostajan että myyjän tiedot on vastaanotettu, rekisteröintipalvelu kutsutaan mukaan prosessiin, jolloin huutokauppatalo syöttää annetut tiedot rekisteröintipalvelulle. Kun palvelu on tallentanut tiedot järjestelmään, se lähettää viestin huutokauppatalolle, joka lopuksi lähettää kiitosviestit ostajalle ja myyjälle merkiksi suorituksen onnistumisesta.

6.3 Prosessin mallinnus

Mallin suunnittelussa lähdetään liikkeelle abstraktilta tasolta miettien mitä prosessin on tarkoitus tehdä, kenen toimesta ja missä järjestyksessä. Siitä edetään askel askeleelta kohti tarkempia yksityiskohtia, eli siirrytään abstraktiotasolla alemmas, jolloin konkreettisuus lisääntyy. Prosessin voidaan ajatella koostuvan moduuleista, joiden sisältö on aluksi hämärän peitossa, mutta ne tarkentuvat suunnittelun edetessä. Kuvassa 27 on hahmoteltu BPD:llä rekisteröintiprosessi korkealla abstraktiotasolla, missä on kuusi erillistä tehtävämoduulia. Kuten jo palvelun kuvauksessa kävi ilmi, prosessissa on neljä osapuolta: ostaja, myyjä, prosessia koordinoiva huutokauppatalo sekä rekisteröintipalvelu.

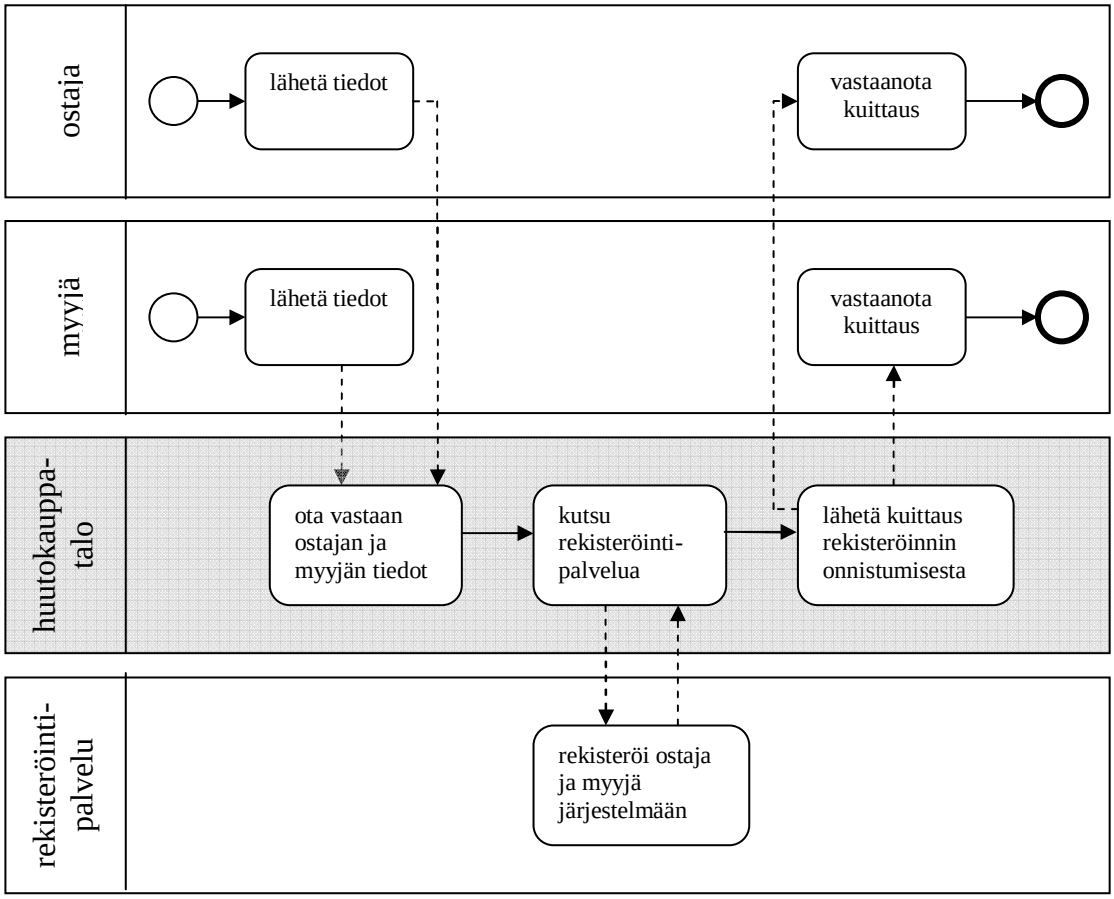
Kuvassa 27 on kaksi katkoviivoilla rajattua aluetta, jotka molemmat liittyvät ostajan ja myyjän kanssa kommunikointiin. Kyseessä on rinnakkaiset tapahtumat, jotka ovat keskenään identtisiä ja niiden suoritusjärjestys on vapaa. Huomioitavaa kuitenkin on, että molemmat tapahtumat täytyy suorittaa kokonaisuudessaan ennen kuin prosessi voi jatkua eteenpäin. Alussa prosessi käynnistyy joko ostajan tai myyjän toimesta kutsumalla huutokauppataloa, jonka hän on WSDL-rajapintakuvauksen avulla löytänyt esimerkiksi UDDI-rekisteristä. Lopussa huutokauppatalo kutsuu ostajaa ja myyjää, ja lähettää heille viestin merkiksi rekisteröinnin onnistumisesta ja prosessin päättymisestä.



Kuva 27. Rekisteröintiprosessin kulku abstraktilla tasolla.

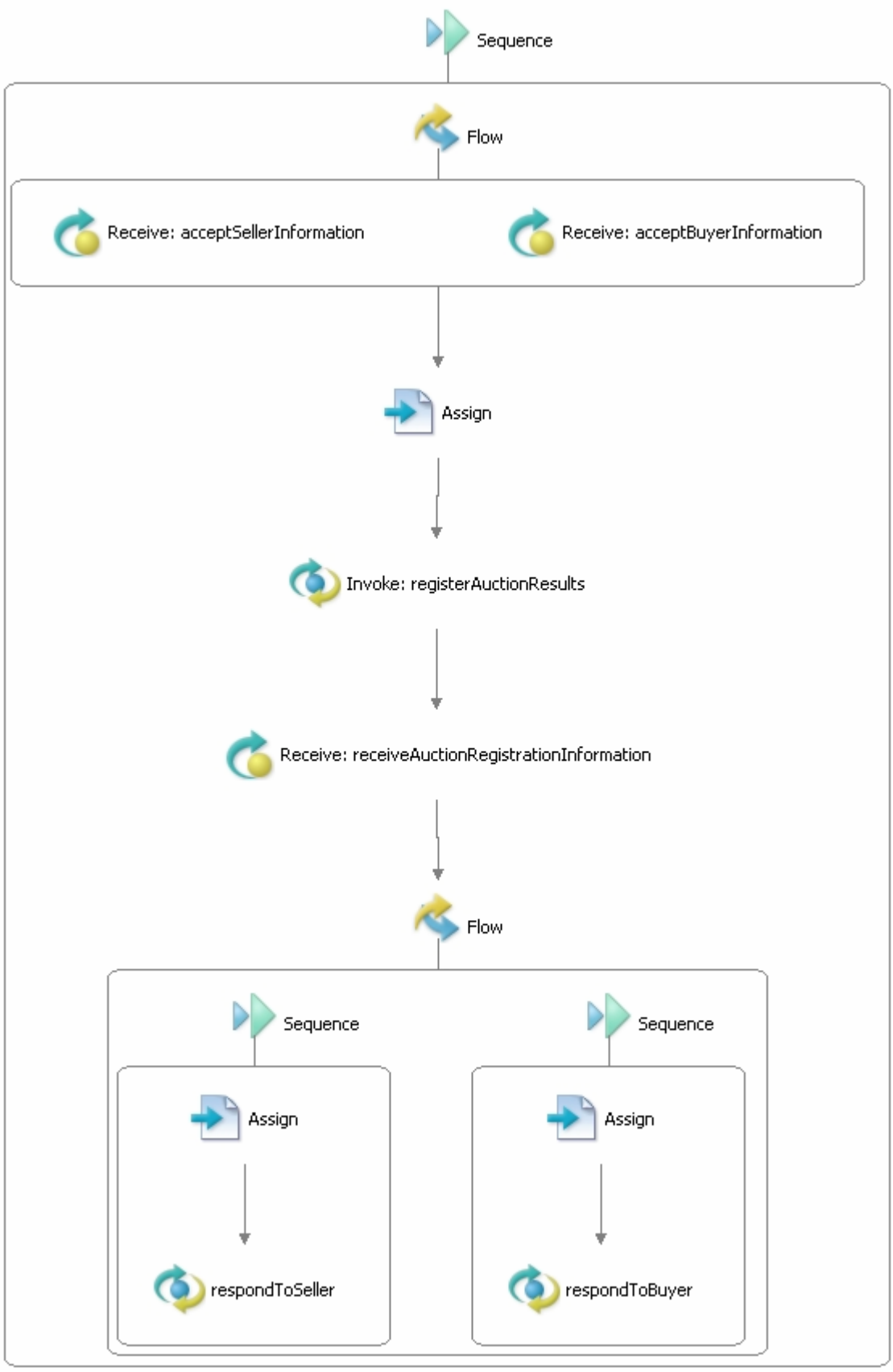
Ennen kuin prosessia voidaan lähteä toteuttamaan ActiveBPEL Designerilla, täytyy vielä selvittää prosessissa tapahtuva viestiliikenne, eli mitä viestejä eri osapuolten välillä liikkuu ja missä järjestyksessä. Myös tähän voidaan käyttää BPD:tä, joka tarjoaa erinomaisen tavan kuvata osapuolten välinen vuorovaikutus uimaratojen avulla. Jokaiselle osapuolelle on omat altaansa, ja prosessi etenee kronologisessa järjestyksessä vasemmalta oikealle kuvan 28 osoittamalla tavalla. Viestivuot kuvaavat osapuolten

välistä vuorovaikutusta, ja ne näkyvät altaiden välissä katkoviivoina. Yhtenäiset nuolet eli sekvenssivuot puolestaan symboloivat aktiviteettien suoritusjärjestystä. Huutokauppatalon allas on kuvassa harmaalla pohjalla, ja kaikki kommunikointi tapahtuu sen kautta. Huutokauppataloon päin tulevat viestivuot edustavat receive-aktiviteettia, poispäin lähtevät viestivuot invoke-aktiviteettia.



Kuva 28. Osapuolten välinen vuorovaikutus voidaan kuvata uima-altaiden avulla.

Kun prosessin osapuolet, tapahtumat ja viestiliikenne on selvillä, on prosessin mallintaminen ActiveBPEL Designerilla melko suoraviivaista. Suurimmat prosessimalleissa näkymättömät operaatiot ovat assign-elementin sisältämät sisäiset kopiointioperaatiot sekä korrelaatiot. Kuvassa 29 on ActiveBPEL Designerista otettu kuvakaappaus valmiista prosessimallista, joka esitellään tarkemmin luvussa 6.4.



Kuva 29. Valmis prosessimalli.

6.4 BPEL-dokumentti

Tähän mennessä prosessi on saatu suunniteltua ja mallinnettua graafisesti, ensin BPMN-notaatiota käyttäen ja sen jälkeen vielä ActiveBPEL Designer mallinnustyökalulla. Graafisista malleista ei kuitenkaan käy ilmi, mitä kuvioiden ja nuolien sisällä konkreettisesti tapahtuu. Seuraavaksi prosessin toiminta käydään läpi yksityiskohtaisesti ActiveBPEL Designerin tuottaman BPEL-koodin avulla, josta selvitetään jokaisen elementin ja ”moduulin” tarkoitus. BPEL-dokumentti löytyy kokonaisuudessaan liitteestä 2.

BPEL on XML-pohjainen kieli, ja BPEL-dokumentti alkaa XML:stä tuttuun tapaan nimiavaruuksien määrittelyllä. Alla olevasta koodista käy ilmi prosessissa käytetyt nimiavaruudet, projektinimenä toimii *auctionhouse*. *suppressJoinFailure* on oletusarvoisesti prosessi-elementin sisällä ”no”, mutta tässä se on määritelty arvoon ”yes”, jolloin prosessi ei pääty virheilmoitukseen joinFailure-virheen ilmetessä. Prosessi-elementin sisällä se toimii ikään kuin globaalin muuttujan tavoin, jolloin se vaikuttaa koko prosessiin, ellei sitä erikseen jossain tietyssä aktiviteetissa muuteta.

```
<bpel:process
xmlns:as="http://example.com/auction/wsdl/auctionService/"
xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="auctionhouse"
suppressJoinFailure="yes" targetNamespace="http://auctionhouse">
```

Jokaisella web-palvelulla on oma WSDL-dokumentti, joka kuvaa palvelun rajapinnan. Myös huutokauppatalo on yksi web-palvelu, joten sekin tarvitsee rajapintakuvauksen. Tässä työssä käytettävä WSDL-dokumentti on nimeltään *auctionServiceInterface.wsdl*, joka tuodaan prosessin käyttöön import-elementin avulla. ActiveBPEL Designer sisältää kätevän *Web References* -toiminnon, joka näyttää WSDL-dokumentissa määritetyt viestit, porttityypit, operaatiot jne. konkreettisina kuvakkeina, joita voidaan suoraan hyödyntää prosessin mallintamisessa.

```
<bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
location="auctionServiceInterface.wsdl"
namespace="http://example.com/auction/wsdl/auctionService/"/>
```

Seuraavaksi määritellään web-palvelujen väliset yhteydet rajapintatasolla <partnerLinks> elementin avulla. Jokaista WSDL-dokumentissa määriteltyä <partnerLinkType> elementtiä kohden tulee vähintään yksi partnerilinkki, tässä tapauksessa yhteensä kolme kappaletta – myyjälle, ostajalle ja rekisteröintipalvelulle kullekin omansa. Koska huutokauppatalo toimii prosessia koordinoivana web-palveluna, asetetaan jokaisen <partnerLinks> elementin *myRole*-attribuutin arvoksi *auctionHouse*.

```
   <bpel:partnerLinks>
      <bpel:partnerLink myRole="auctionHouse" name="seller"
partnerLinkType="as:sellerAuctionHouseLT" partnerRole="seller"/>
      <bpel:partnerLink myRole="auctionHouse" name="buyer"
partnerLinkType="as:buyerAuctionHouseLT" partnerRole="buyer"/>
      <bpel:partnerLink myRole="auctionHouse"
name="auctionRegistrationService"
partnerLinkType="as:auctionHouseAuctionRegistrationServiceLT"
partnerRole="auctionRegistrationService"/>
   </bpel:partnerLinks>
```

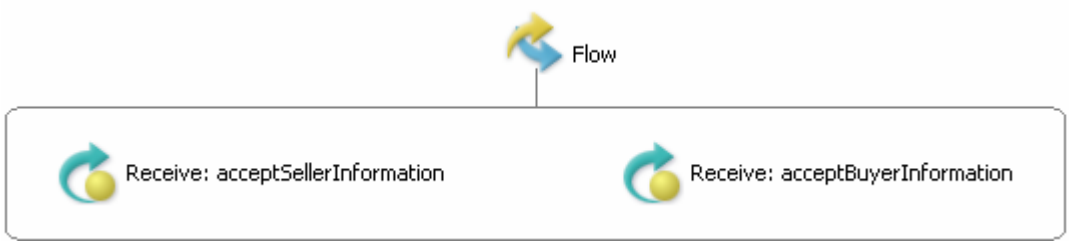
Ennen varsinaisen prosessin alkua määritellään myös tarvittavat muuttujat, joita tässä tapauksessa on kuusi; myyjälle, ostajalle ja huutokaupalle kullekin kaksi muuttujaa. *Answer*-muuttujia käytetään vain vastausviestien välittämiseen palvelua pyytäville osapuolille.

```
<bpel:variables>
   <bpel:variable messageType="as:sellerData" name="sellerData"/>
   <bpel:variable messageType="as:sellerAnswerData"
name="sellerAnswerData"/>
   <bpel:variable messageType="as:buyerData" name="buyerData"/>
   <bpel:variable messageType="as:buyerAnswerData"
name="buyerAnswerData"/>
   <bpel:variable messageType="as:auctionData" name="auctionData"/>
   <bpel:variable messageType="as:auctionAnswerData"
name="auctionAnswerData"/>
</bpel:variables>
```

Prosessissa tarvitaan myös korrelaatiota tunnistamaan olemassa olevia prosessi-instansseja, tässä tapauksessa huutokaupan tunnistenumeron *auctionId:n* perusteella. Kun esimerkiksi myyjän lähettämän viestin laukaisemana syntyy uusi prosessi-instanssi, ei ostajan viesti luo toista erillistä instanssia, vaan se korreloidaan tunnistenumeron avulla aiemmin luotuun instanssiin.

```
<bpel:correlationSets>
   <bpel:correlationSet name="auctionIdentification"
properties="as:auctionId"/>
</bpel:correlationSets>
```

Nimiavaruuksien, partnerilinkkien, muuttujien ja korrelaatioiden määrittelyjen jälkeen päästään varsinaiseen prosessia suorittavaan osaan BPEL-dokumentissa. Prosessi alkaa siis myyjän tai ostajan kutsusta, sillä ei ole väliä kumpi huutokauppataloa ensin kutsuu. Kun kyseessä on rinnakkainen tapahtuma, molemmat <receive> elementit kapseloidaan <flow> elementin sisään (kuva 30). Tässä yhteydessä tarvitaan korrelaatioita, jotta ei synny kahta erillistä prosessi-instanssia. Uusi prosessi-instanssi syntyy siinä vaiheessa, kun ensimmäisenä huutokauppaa kutsuvan osapuolen viesti vastaanotetaan, jonka jälkeen toisen osapuolen viestit ohjataan korrelaation avulla suoraan olemassa olevalle instanssille. Tätä varten korrelaatioon on määritelty attribuutiksi *initiate=”join”*, eli ennen uuden instanssin luomista tutkitaan löytyykö valmista instanssia johon liittyä.



Kuva 30. Prosessin käynnistyminen.

Kaikki prosessin toiminnalliset osat on kapseloitu <sequence> elementin sisään, ja sen vaikutus kestää prosessin loppuun asti. *Sequence* tarkoittaa sitä, että aktiviteetit suoritetaan määrätyssä järjestyksessä, ja esimerkiksi edellä kuvatussa rinnakkaisessa tapahtumassa prosessi jatkuu vasta, kun sekä myyjän että ostajan tiedot on vastaanotettu.

```
<bpel:sequence>

<bpel:flow>

<bpel:receive createInstance="yes" name="acceptSellerInformation"
operation="submit" partnerLink="seller" portType="as:sellerPT"
variable="sellerData">

<bpel:correlations>
```

```
   <bpel:correlation initiate="join" set="auctionIdentification"/>
</bpel:correlations>

</bpel:receive>

<bpel:receive createInstance="yes" name="acceptBuyerInformation"
operation="submit" partnerLink="buyer" portType="as:buyerPT"
variable="buyerData">

<bpel:correlations>
   <bpel:correlation initiate="join" set="auctionIdentification"/>
</bpel:correlations>

</bpel:receive>

</bpel:flow>
```

Ennen rekisteröintipalvelun kutsumista täytyy ulospäin lähtevän viestin muuttujille asettaa arvot kopiointioperaatioilla. WSDL-dokumentissa määritelty *auctionData*-viesti koostuu kuudesta osasta: integer tyyppisistä *auctionId* ja *amount* -muuttujista, ostajalta tulevista *buyerCreditCardNumber* ja *phoneNumber* -muuttujista sekä myyjän *sellerCreditCardNumber* ja *shippingCosts* -muuttujista. Arvojen asettamiseen käytetään <assign> elementtiä. *Amount*-muuttujaan asetetaan tässä tapauksessa staattinen kokonaisluku, mutta sekin voitaisiin kysyä myyjältä muiden tietojen ohella.

```
<bpel:assign>
         <bpel:copy>
            <bpel:from part="auctionId" variable="sellerData"/>
            <bpel:to part="auctionId" variable="auctionData"/>
         </bpel:copy>
         <bpel:copy>
            <bpel:from>1</bpel:from>
            <bpel:to part="amount" variable="auctionData"/>
         </bpel:copy>
         <bpel:copy>
            <bpel:from part="creditCardNumber" variable="buyerData"/>
            <bpel:to part="buyerCreditCardNumber"
variable="auctionData"/>
         </bpel:copy>
         <bpel:copy>
            <bpel:from part="phoneNumber" variable="buyerData"/>
            <bpel:to part="phoneNumber" variable="auctionData"/>
         </bpel:copy>
         <bpel:copy>
            <bpel:from part="creditCardNumber" variable="sellerData"/>
            <bpel:to part="sellerCreditCardNumber"
variable="auctionData"/>
         </bpel:copy>
         <bpel:copy>
            <bpel:from part="shippingCosts" variable="sellerData"/>
            <bpel:to part="shippingCosts" variable="auctionData"/>
```

```
          </bpel:copy>
      </bpel:assign>
```

Kun huutokauppatalo on tehnyt sisäiset kopiointioperaationsa, on aika kutsua rekisteröintipalvelu mukaan prosessiin. Partnerilinkkinä on tällöin *auctionRegistrationService* ja porttityyppinä *auctionRegistrationPT*. Rekisteröintipalvelun suoritettua tehtävänsä se kutsuu huutokauppataloa, joka ottaa vastaan *auctionRegistrationAnswerPT*-porttityypin kautta tiedon rekisteröinnin onnistumisesta ja tallettaa sen *auctionAnswerData* nimiseen muuttujaan. Myös tässä tarvitaan korrelaatiota, jotta rekisteröintipalvelulta tuleva vastausviesti menee oikeaan osoitteeseen.

```
<bpel:invoke inputVariable="auctionData" name="registerAuctionResults"
operation="process" partnerLink="auctionRegistrationService"
portType="as:auctionRegistrationPT"/>

<bpel:receive name="receiveAuctionRegistrationInformation"
operation="answer" partnerLink="auctionRegistrationService"
portType="as:auctionRegistrationAnswerPT"
variable="auctionAnswerData">

<bpel:correlations>
   <bpel:correlation initiate="no" set="auctionIdentification"/>
</bpel:correlations>

</bpel:receive>
```

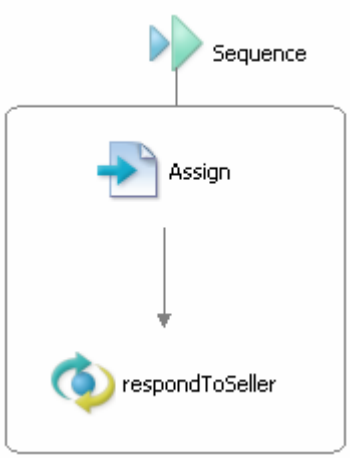
Rekisteröintipalvelu näkyy prosessissa ikään kuin mustana laatikkona, jonka rajapinta tunnetaan mutta sisäistä toimintaa ei. Kuvassa 31 on näkymä edellä kuvatusta tapahtumasarjasta, joka koostuu huutokauppatalon näkökulmasta yhdestä <invoke> ja yhdestä <receive> aktiviteetista.

Kuva 31. Rekisteröintipalvelun kutsuminen ja palvelun vastaus.

Tietojen rekisteröimisen jälkeen on jäljellä enää kiitosviestien lähettäminen ostajalle ja myyjälle. Koska kyseessä on kaksi rinnakkain tapahtuvaa operaatiota, ne kapseloidaan <flow> elementin sisään. Operaatiot ovat identtisiä muuttujan nimeä lukuun ottamatta, ja alla olevassa koodissa on esimerkin vuoksi vain myyjän kapseli. *sellerAnswerData* nimiseen muuttujaan kopioidaan teksti ”*Thank You!*”, jonka jälkeen kutsutaan myyjää ja kyseinen viesti lähetetään hänelle. Se on samalla merkki rekisteröinnin onnistumisesta ja prosessin päättymisestä. Käytännössä prosessi päättyy siinä vaiheessa kun kiitosviesti on lähetetty molemmille osapuolille, ja kaikki avoimet elementit suljetaan, viimeisenä luonnollisesti <process> elementti.

```
<bpel:flow>

<bpel:sequence>
   <bpel:assign>
      <bpel:copy>
         <bpel:from>
            <bpel:literal>Thank You!</bpel:literal>
         </bpel:from>
            <bpel:to>$sellerAnswerData.thankYouText</bpel:to>
      </bpel:copy>

   </bpel:assign>
   <bpel:invoke inputVariable="sellerAnswerData"
name="respondToSeller" operation="answer" partnerLink="seller"
portType="as:sellerAnswerPT"/>

</bpel:sequence>

</bpel:flow>

</bpel:sequence>

</bpel:process>
```

Edellä kuvattu <flow> elementti sisältää sekä myyjälle että ostajalle omat sequence-kapselit, jotta kaikki tarvittavat operaatiot tulevat suoritettua oikeassa järjestyksessä ennen kiitosviestien lähettämistä. Kuvassa 32 on esitetty myyjää koskeva kapseli, ostajan kapseli on sen kanssa identtinen.



Kuva 32. Myyjää koskeva sequence-kapseli kiitosviestin lähettämistä varten.

6.5 Rekisteröintipalvelun rajapinta

WSDL-dokumentti alkaa BPEL-dokumentin tavoin nimiavaruuksien määrittelyllä. Ne tulevat heti <definitions> elementin jälkeen, joka on WSDL-dokumentin juurielementti. Attribuutilla *targetNamespace* dokumentin määrittelyt voidaan sijoittaa haluttuun nimiavaruuteen, muut työssä käytetyt nimiavaruudet ovat virallisia WS-BPEL standardin suosituksia.

```
<wsdl:definitions
   targetNamespace="http://example.com/auction/wsdl/auctionService/"
   xmlns:bpel="http://docs.oasis-
open.org/wsbpel/2.0/process/executable"
   xmlns:vprop="http://docs.oasis-open.org/wsbpel/2.0/varprop"
   xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
   xmlns:tns="http://example.com/auction/wsdl/auctionService/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Viestit koostuvat osista, ja jokainen osa koostuu kahdesta attribuutista; nimestä ja tyypistä. Tyyppiattribuutti voi olla joko *type* tai *element*, tässä tapauksessa kaikki osat sisältävät *type*-attribuutin. *Name*-attribuutti määrittelee yksiselitteisen nimen sekä viestille että viestin osalle. Kullakin viestillä voi olla useampi osa, esimerkiksi *sellerData* koostuu kolmesta ja *auctionData* kuudesta osasta, sillä sen täytyy välittää kaikki ostajan ja myyjän antamat tiedot.

Kaksi ensimmäistä viestiä, *sellerData* ja *sellerAnswerData*, ovat kuvauksia myyjän ja huutokauppatalon välisestä kommunikoinnista, *buyerData* ja *buyerAnswerData* puolestaan ostajan ja huutokauppatalon keskisestä kommunikoinnista. Sekä myyjältä että ostajalta kysytään luottokortinnumero, lisäksi toimituskulut ja ostajan puhelinnumero tallennetaan järjestelmään. Huutokauppatalon ja rekisteröintipalvelun välistä viestintää määrittelevät *auctionData* ja *auctionAnswerData* -viestit, missä *registrationId* on tunnistenumero rekisteröidylle tiedolle.

```
<wsdl:message name="sellerData">
   <wsdl:part name="creditCardNumber" type="xsd:string" />
   <wsdl:part name="shippingCosts" type="xsd:integer" />
   <wsdl:part name="auctionId" type="xsd:integer" />
</wsdl:message>

<wsdl:message name="sellerAnswerData">
   <wsdl:part name="thankYouText" type="xsd:string" />
</wsdl:message>

<wsdl:message name="buyerData">
   <wsdl:part name="creditCardNumber" type="xsd:string" />
   <wsdl:part name="phoneNumber" type="xsd:string" />
   <wsdl:part name="auctionId" type="xsd:integer" />
</wsdl:message>

<wsdl:message name="buyerAnswerData">
   <wsdl:part name="thankYouText" type="xsd:string" />
</wsdl:message>

<wsdl:message name="auctionData">
   <wsdl:part name="auctionId" type="xsd:integer" />
   <wsdl:part name="amount" type="xsd:integer" />
   <wsdl:part name="sellerCreditCardNumber" type="xsd:string" />
   <wsdl:part name="shippingCosts" type="xsd:integer" />
   <wsdl:part name="buyerCreditCardNumber" type="xsd:string" />
   <wsdl:part name="phoneNumber" type="xsd:string" />
</wsdl:message>

<wsdl:message name="auctionAnswerData">
   <wsdl:part name="registrationId" type="xsd:integer" />
   <wsdl:part name="auctionId" type="xsd:integer" />
</wsdl:message>
```

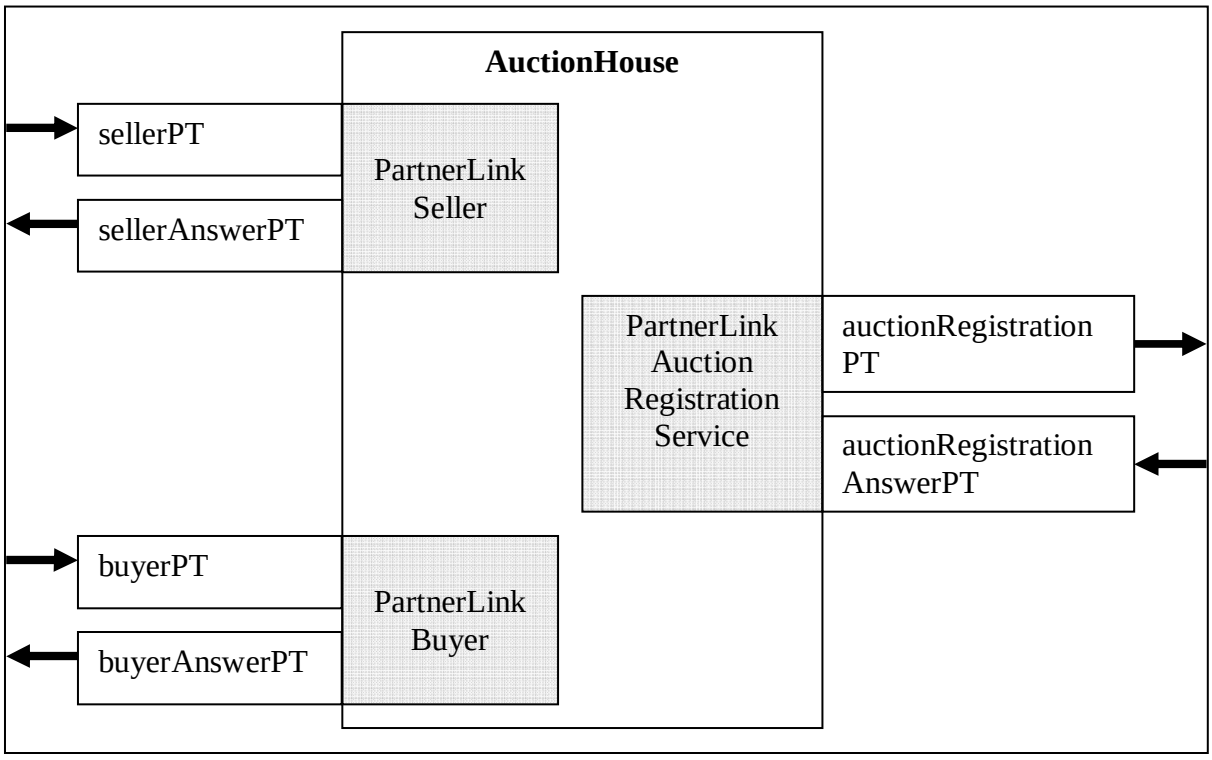
Porttityyppi on kokoelma abstrakteja toimintoja sekä niihin liittyviä viestejä. Toiminnot määritellään operaatio-elementillä, esimerkiksi *sellerPT:n* toiminto on *submit*. Toimintoja on neljää eri tyyppiä: yksisuuntainen, pyyntö-vastaus, lähetys-vastaus ja ilmoitus. Tässä tapauksessa jokaisella porttityypillä on pelkästään *input*-elementti, eli ne

ovat yksisuuntaisia toimintotyyppejä. Toiminnot viittaavat viesteihin *message*-attribuutin kautta, esimerkiksi *sellerPT*-porttityypin toiminto viittaa *sellerData*-viestiin.

Myyjän ja ostajan lähettämät tiedot kirjautuvat palveluun kahden porttityypin, *sellerPT* ja *buyerPT*, sekä sopivien operaatioiden ansiosta. Palvelu vastaa myyjälle ja ostajalle puolestaan *sellerAnswerPT* ja *buyerAnswerPT* -porttityyppien kautta. *sellerAnswerPT* ja *buyerAnswerPT* -porttityypit liittyvät kahteen <partnerLinkType> elementtiin, *sellerAuctionHouseLT* ja *buyerAuctionHouseLT*. Vastaavat porttityypit rekisteröintipalvelun suuntaan ovat *auctionRegistrationPT* ja *auctionRegistrationAnswerPT*, <partnerLinkType> elementti on nimeltään *auctionHouseAuctionRegistrationServiceLT*.

```
<wsdl:portType name="sellerPT">
   <wsdl:operation name="submit">
      <wsdl:input message="tns:sellerData" />
   </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="sellerAnswerPT">
   <wsdl:operation name="answer">
      <wsdl:input message="tns:sellerAnswerData" />
   </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="buyerPT">
   <wsdl:operation name="submit">
      <wsdl:input message="tns:buyerData" />
   </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="buyerAnswerPT">
   <wsdl:operation name="answer">
      <wsdl:input message="tns:buyerAnswerData" />
   </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="auctionRegistrationPT">
   <wsdl:operation name="process">
      <wsdl:input message="tns:auctionData" />
   </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="auctionRegistrationAnswerPT">
   <wsdl:operation name="answer">
      <wsdl:input message="tns:auctionAnswerData" />
   </wsdl:operation>
</wsdl:portType>
```

Kuvassa 33 on havainnollistettu porttityyppien ja partnerilinkkien yhteyttä toisiinsa sekä niiden roolia viestien välityksessä osapuolten välillä. Tässä jokaiseen partnerilinkkiin liittyy kaksi porttityyppiä, sisäänpäin tuleville sekä ulospäin lähteville viesteille omansa.



Kuva 33. Rekisteröintiprosessissa tarvitaan kolmea partnerilinkkiä sekä kuutta porttityyppiä.

Aliaksia käytetään saapuvien viestien korreloinnissa, jolloin huutokaupan tunnistenumeron *auctionId:n* perusteella liiketoimintaprosessi voidaan paikallistaa. <propertyAlias> viittaa aina johonkin viestin osaan, tässä kunkin viestin *auctionId*-osaan, jotka assosioidaan <property> elementin *name*-attribuutissa määriteltyyn *auctionId*-nimeen.

```
<vprop:property name="auctionId" type="xsd:integer" />
<vprop:propertyAlias propertyName="tns:auctionId"
   messageType="tns:sellerData" part="auctionId" />
<vprop:propertyAlias propertyName="tns:auctionId"
   messageType="tns:buyerData" part="auctionId" />
<vprop:propertyAlias propertyName="tns:auctionId"
   messageType="tns:auctionData" part="auctionId" />
<vprop:propertyAlias propertyName="tns:auctionId"
   messageType="tns:auctionAnswerData" part="auctionId" />
```

<partnerLinkType> elementti määrittää web-palvelujen välisen vuorovaikutuksen roolit, jotta palvelut ymmärtävät vaihdettavien viestien kontekstin. Yksi <partnerLinkType> koostuu (yleensä) kahdesta roolista, joista kumpikin liittyy tasan yhteen porttityyppiin. Esimerkiksi ensimmäinen, *sellerAuctionHouseLT*, kuvaa myyjän ja huutokauppatalon välisen suhteen, *buyerAuctionHouseLT* ostajan ja huutokauppatalon välisen suhteen jne. Jokaista web-palveluparia ja interaktiota kohden on määriteltävä oma <partnerLinkType> elementti.

```
   <plnk:partnerLinkType name="sellerAuctionHouseLT">
      <plnk:role name="auctionHouse" portType="tns:sellerPT" />
      <plnk:role name="seller" portType="tns:sellerAnswerPT" />
   </plnk:partnerLinkType>

   <plnk:partnerLinkType name="buyerAuctionHouseLT">
      <plnk:role name="auctionHouse" portType="tns:buyerPT" />
      <plnk:role name="buyer" portType="tns:buyerAnswerPT" />
   </plnk:partnerLinkType>

   <plnk:partnerLinkType
      name="auctionHouseAuctionRegistrationServiceLT">
      <plnk:role name="auctionRegistrationService"
         portType="tns:auctionRegistrationPT" />
      <plnk:role name="auctionHouse"
         portType="tns:auctionRegistrationAnswerPT" />
   </plnk:partnerLinkType>

<wsdl:service name="auctionServiceBP"/>

</wsdl:definitions>
```

Ennen juurielementin sulkemista WSDL-dokumentissa on vielä <service> elementti, joka normaalisti määrittelee joukon portteja. Portit ovat web-palvelujen liittymäpisteitä verkkoon, sillä jokaisella portilla on oma porttityyppi, sidos ja yksiselitteinen osoite. Nyt, kun todellisia web-palveluja ei käytetä, ei porttejakaan tarvitse määritellä. Palvelun nimeksi on annettu *auctionServiceBP*.

6.6 Prosessin testaus

Mallinnetun prosessin testaus suoritetaan niin ikään ActiveBPEL Designerin avulla. Ennen kuin simulointia on mahdollista aloittaa, täytyy varmistua siitä, ettei prosessia

tulkittaessa löydy yhtään virhettä. Mikäli virheitä esiintyy, ei ohjelma suostu käynnistämään simulointia ennen kuin virheet on korjattu.

Kun prosessista luotu BPEL-koodi on määritysten mukaista eikä virheitä enää esiinny, voidaan muuttujille antaa esimerkkisisältöjä, joilla prosessia simuloidaan. Esimerkkisisällöt tallennetaan *sellerDataan* taulukon 1 ja *buyerDataan* taulukon 2 mukaisesti. *AuctionData*-viestin sisältö täydentyy automaattisesti prosessin aikana, joten sille ei simulointivaiheessakaan tarvitse antaa arvoja.

Muuttuja	**Tyyppi**	**Arvo**
creditCardNumber	string	22334455
shippingCosts	integer	50
auctionId	integer	1

Taulukko 1. Esimerkkisisällöt *sellerData*-tyyppisen viestin muuttujille.

Muuttuja	**Tyyppi**	**Arvo**
creditCardNumber	string	15253545
phoneNumber	string	0501234567
auctionId	integer	1

Taulukko 2. Esimerkkisisällöt *buyerData*-tyyppisen viestin muuttujille.

Edellä lueteltujen lisäksi myyjällä, ostajalla ja huutokaupalla on omat muuttujansa paluuviestille, joista kaksi ensimmäistä eli *sellerAnswerData* ja *buyerAnswerData* sisältävät vain yhden muuttujan nimeltään *thankYouText*. Näistä kumpikin saa prosessin edetessä arvon *”Thank You!”*, joka on tyypiltään luonnollisesti merkkijono. Sen sijaan *auctionAnswerData* koostuu kahdesta muuttujasta, kokonaislukuja ymmärtävistä *auctionId* ja *registrationId* -muuttujista. Taulukosta 3 selviää *auctionAnswerDatan* esimerkkisisällöt.

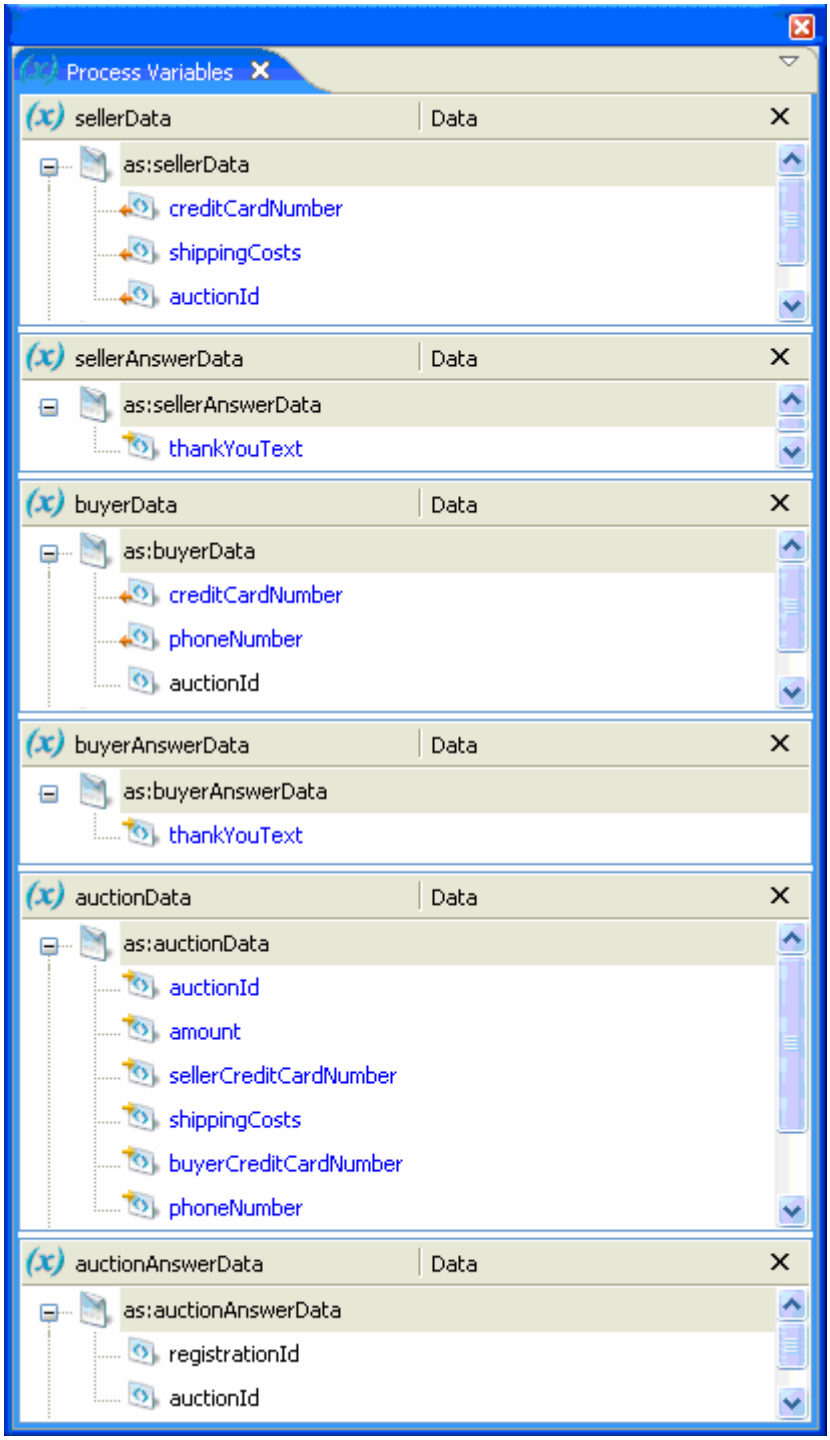
Muuttuja	Tyyppi	Arvo
auctionId	integer	1
registrationId	integer	1

Taulukko 3. Esimerkkisisällöt *auctionAnswerData*-tyyppisen viestin muuttujille.

Prosessin simulointi aloitetaan painamalla työkalupalkista löytyvää *Start Simulation* -nappia, jolloin virtuaalinen työnkulkukone käynnistyy ja ActiveBPEL Designer siirtyy virheentarkastustilaan. Samalla luodaan uusi prosessi-instanssi sekä tyhjennetään muuttujien sisältö, kuten kuva 34 havainnollistaa. Sitä mukaa kun prosessi etenee, muuttujat saavat arvoja aivan kuin ajettaisiin todellista web-palvelua.

Prosessi voidaan testata askel askeleelta, ja ohjelma korostaa kulloinkin suoritusvaiheessa olevan aktiviteetin sinisellä ympyrällä. Rekisteröintiprosessin ensimmäinen askel on sequence-elementti, josta edetään flow-elementin sisältämiin receive-aktiviteetteihin. Tässä vaiheessa saadaan ensimmäiset muuttujien arvot asetettua, kun ostaja ja myyjä lähettävät tietonsa huutokauppatalolle.

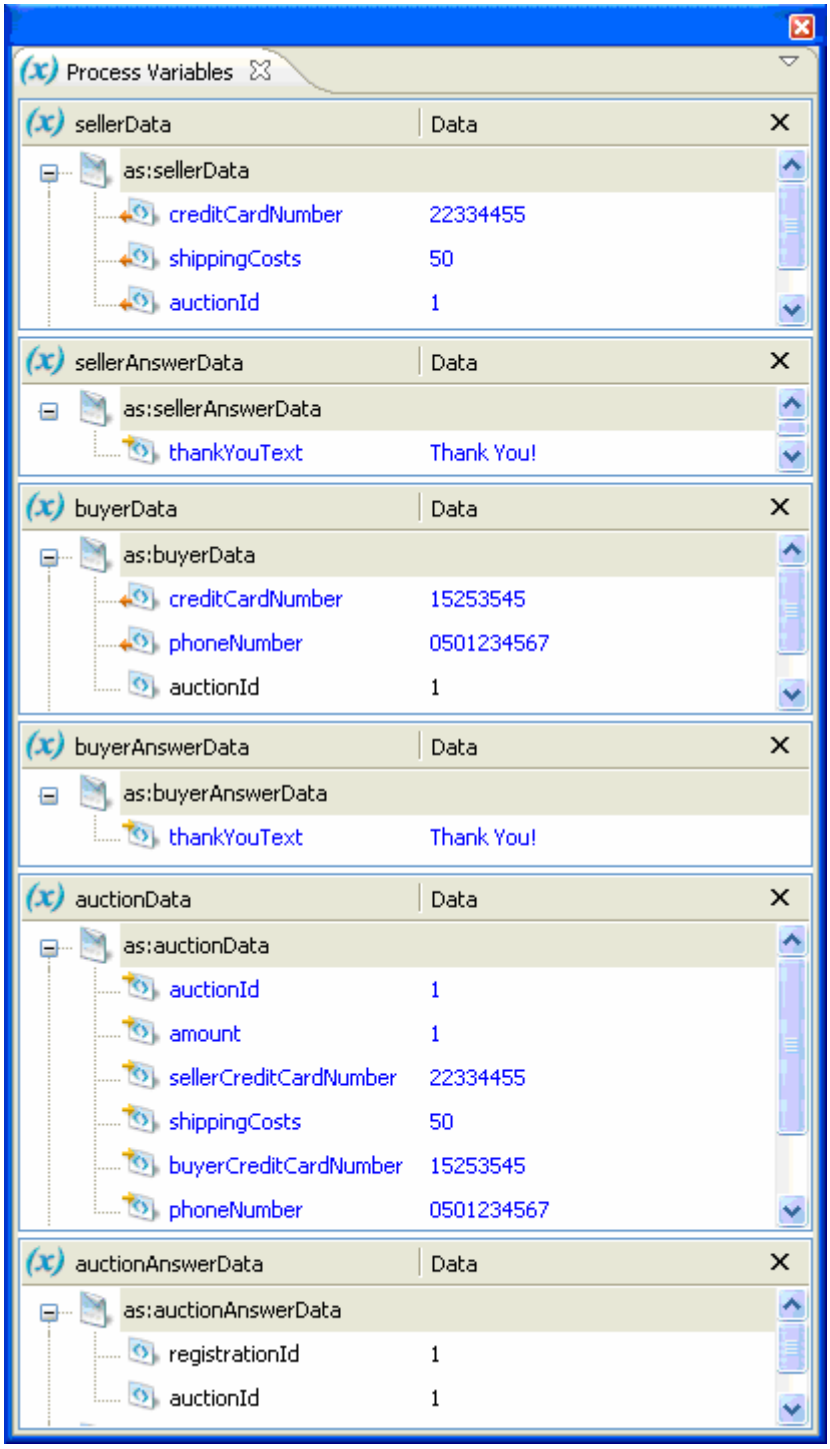
Seuraava vaihe on assign-elementin sisäisten kopiointioperaatioiden suorittaminen, jolloin *auctionData*-tyyppisen viestin muuttujille asetetaan sisältö. Tämän jälkeen voidaan kutsua rekisteröintipalvelua, joka simulointia ajettaessa kuvitteellisesti rekisteröi osapuolet järjestelmään ja lähettää *auctionAnswerData*-tyyppisen viestin huutokauppatalolle. Samalla *auctionId* ja *registrationId* -muuttujat saavat uuden arvon.



Kuva 34. Simuloinnin alkaessa muuttujat eivät sisällä arvoja.

Tässä vaiheessa muut muuttujat ostajan ja myyjän *thankYouText*-muuttujaa lukuun ottamatta ovat jo saaneet uuden arvon. Flow-elementin kautta siirrytään jompaankumpaan - joko ostajan tai myyjän - sequence-elementtiin, ja sen sisältämään assign-elementtiin. Kiitosviesti kopioidaan *thankYouText*-muuttujaan ja viesti lähetetään

vastaanottajalle. Jäljellä on enää samojen operaatioiden suorittaminen toiselle osapuolelle, jonka jälkeen kaikki avoimet elementit suljetaan ja prosessi päättyy. Kuva 35 on simuloinnin päättymisen jälkeen otettu kuvakaappaus, josta voidaan havaita kaikkien muuttujien saaneen tietynlaisen arvon (vrt. kuva 34).



Kuva 35. Simuloinnin päättyessä kaikille muuttujille on asetettu arvot.

ActiveBPEL Designer tuottaa simuloinnin tuloksena myös prosessilokia, joka löytyy kokonaisuudessaan alla olevasta listauksesta. Kaikki prosessin tapahtumat kirjautuvat aloitus- ja lopetusmerkintöineen lokiin, ja sen avulla esimerkiksi virheiden jäljittäminen helpottuu. Virhetilanteissa lokiin tallentuu punaisella fontilla yksityiskohtaista tietoa siitä, missä virhe tapahtui ja mistä se johtui.

```
Process auctionhouse: Instance 1 created.
Process auctionhouse: Executing [/process]
Process Suspended [/process]
Sequence : Executing [/process/sequence]
Flow : Executing [/process/sequence/flow]
Receive acceptSellerInformation: Executing
[/process/sequence/flow/receive[@name='acceptSellerInformation']]
Receive acceptSellerInformation: Completed normally
[/process/sequence/flow/receive[@name='acceptSellerInformation']]
Receive acceptBuyerInformation: Executing
[/process/sequence/flow/receive[@name='acceptBuyerInformation']]
Receive acceptBuyerInformation: Completed normally
[/process/sequence/flow/receive[@name='acceptBuyerInformation']]
Flow : Completed normally [/process/sequence/flow]
Assign : Executing [/process/sequence/assign]
Assign : Completed normally [/process/sequence/assign]
Invoke registerAuctionResults: Executing
[/process/sequence/invoke[@name='registerAuctionResults']]
Invoke registerAuctionResults: Completed normally
[/process/sequence/invoke[@name='registerAuctionResults']]
Receive receiveAuctionRegistrationInformation: Executing
[/process/sequence/receive[@name='receiveAuctionRegistrationInformatio
n']]
Receive receiveAuctionRegistrationInformation: Completed normally
[/process/sequence/receive[@name='receiveAuctionRegistrationInformatio
n']]
Flow : Executing [/process/sequence/flow[2]]
Sequence : Executing [/process/sequence/flow[2]/sequence]
Sequence : Executing [/process/sequence/flow[2]/sequence[2]]
Assign : Executing [/process/sequence/flow[2]/sequence/assign]
Assign : Completed normally
[/process/sequence/flow[2]/sequence/assign]
Assign : Executing [/process/sequence/flow[2]/sequence[2]/assign]
Assign : Completed normally
[/process/sequence/flow[2]/sequence[2]/assign]
Invoke respondToSeller: Executing
[/process/sequence/flow[2]/sequence/invoke[@name='respondToSeller']]
Invoke respondToSeller: Completed normally
[/process/sequence/flow[2]/sequence/invoke[@name='respondToSeller']]
Sequence : Completed normally [/process/sequence/flow[2]/sequence]
Invoke respondToBuyer: Executing
[/process/sequence/flow[2]/sequence[2]/invoke[@name='respondToBuyer']]
Invoke respondToBuyer: Completed normally
[/process/sequence/flow[2]/sequence[2]/invoke[@name='respondToBuyer']]
Sequence : Completed normally [/process/sequence/flow[2]/sequence[2]]
Flow : Completed normally [/process/sequence/flow[2]]
Sequence : Completed normally [/process/sequence]
Process auctionhouse: Completed normally [/process]
```

Tässä työssä toteutettu prosessi vaatii, että kaikki vaadittavat tiedot on annettu, ja että ne ovat oikean tyyppisiä. Prosessin virhesietoisuutta voitaisiin lisätä virheiden käsittelijällä, joka esimerkiksi *auctionId:n* puuttuessa kehottaisi ostajaa tai myyjää antamaan kaikki vaadittavat tiedot. Virheiden käsittelijää voitaisiin käyttää myös siinä tilanteessa, että rekisteröinti ei jostain syystä onnistuisi, ja se lähettäisi ostajalle ja myyjälle viestin rekisteröinnin epäonnistumisesta.

Vaikka prosessin testauksessa ei käytettykään erillistä olemassa olevaa web-palvelua, osoittaa simuloinnin onnistunut läpimeno ja muuttujien arvojen oikeanlainen käyttäytyminen prosessin toimivan oletetulla tavalla. Testaus suoritettiin useilla eri esimerkkisyötteillä ja joka kerta prosessi suoritti tehtävät oikein ja oikeassa järjestyksessä. Näin ollen voidaan olettaa, että mikäli ulkoinen rekisteröintipalvelu suoriutuu tehtävästään kunniallisesti, toimii työssä toteutettu rekisteröintiprosessi myös käytännön tasolla.

Seuraava vaihe rekisteröintiprosessin kehityksessä ja testauksessa olisi käytännössä toteuttaa tiedot rekisteröivä web-palvelu. Rekisteröintipalvelun rajapinnassa olisi huomioitava sopiva saapuvien ja lähtevien viestien muoto, jotta kommunikointi huutokauppatalon kanssa onnistuisi. Tämän jälkeen prosessia voitaisiin testata konkreettisesti ActiveBPEL-koneella, joka vaatii toimiakseen Apache Tomcatin tms. sekä Java Development Kitin. Prosessin hallinta tapahtuisi helpoiten komentokehotteesta käsin, tai toinen vaihtoehto olisi toteuttaa lomakepohjainen web-sivu, johon käyttäjä voisi syöttää tarvittavat tiedot.

6.7 Lopputulos ja johtopäätökset

Työn tarkoituksena oli tutkia työnkulkukoneen, ja erityisesti ActiveBPEL-koneen soveltuvuutta web-palvelupohjaisen sähköisen huutokaupan moottorina. Tutkimus suoritettiin mallintamalla ja simuloimalla yksi osa huutokauppajärjestelmää, jonka pohjalta olisi mahdollista suunnitella ja toteuttaa kokonainen järjestelmä. Tämän työn periaatteiden mukaista täysimittaista sähköistä huutokauppaa ei kirjoitushetkellä ole

olemassa, ja tulosten hyödyntäminen liiketoiminnassa vaatii aiheesta lisää tutkimusta sekä yritysten kiinnostusta web-palvelupohjaisia ratkaisuja kohtaan.

Tutkimuksen näkyvimpänä tuloksena syntyi ActiveBPEL Designer -ohjelmalla toteutettu prosessimalli, sekä ohjelman automaattisesti tuottama BPEL-koodi prosessimallista. Myös simuloinnin tuloksena syntynyt prosessiloki kertoo arvokasta tietoa prosessin toiminnasta. WSDL-rajapintakuvauksen runko on peräisin OASIS-organisaation WS-BPEL 2.0 -standardin verkkosivuilta löytyvästä *Auction service* esimerkistä, mutta tutkimuksen aikana siihen tehtyjen lukuisten muutosten seurauksena voidaan WSDL-dokumenttiakin pitää tutkimuksen tuloksena syntyneenä sivutuotteena.

Ennen varsinaista toteutusvaihetta prosessi mallinnettiin abstraktilla tasolla BPMN:n avulla. Korkean tason malleista on hyötyä myöhemmässä vaiheessa, sillä kokonaisuuden hahmottaminen ja prosessin teoreettinen suunnittelu on välttämätöntä hyvän lopputuloksen aikaansaamiseksi. Esimerkiksi uima-altaiden suunnittelu edesauttaa eri osasten harmoniaa heti alusta lähtien, kun mallia varten joudutaan pohtimaan osapuolten välisiä vuorovaikutuksia prosessin aikana. Vanha sanonta, jonka mukaan hyvin suunniteltu on puoliksi tehty, pätee erinomaisesti tässäkin tapauksessa. Lisäksi BPMN-malleja on hyvin usein helppo muuttaa suoraan BPEL4WS-määritykseksi, jolloin malleista saatava hyöty korostuu entisestään. Esimerkkinä luvussa 6.3 esitellystä abstraktista mallista (kuva 27) on löydettävissä paljon yhteneväisyyksiä valmiin BPEL-prosessimallin (kuva 29) kanssa.

Yksi työn keskeisimmistä tavoitteista oli rekisteröintiprosessin toteuttaminen. Järkevältä näyttävä prosessimalli ja systemaattisesti oikein toimiva prosessisimulaatio osoittavat tavoitteen onnistumisesta niiltä osin. Toteutettuun prosessiin jouduttiin kuitenkin tekemään joitakin kompromisseja tutkimuksen aikana, sillä konkreettisten web-palvelujen puuttuessa dynaamisuus jäi vähäiseksi. Prosessista puuttuu muun muassa kokonaan sidokset ja portit, joilla määritellään käytettävät tietoliikenneprotokollat ja tiedonsiirtoformaatit. Abstraktista luonteestaan huolimatta mallinnettu prosessi hoitaa tehtävänsä tutkimusongelmaa ratkaisevana tutkimusmenetelmänä, eli se auttaa

vastaamaan kysymykseen: ”*Miten ActiveBPEL soveltuu sähköisen huutokaupan moottoriksi?*”.

Tutkimusongelman vastaukseen vaikuttaa luonnollisesti useita eri tekijöitä. Jos ajatellaan sähköisen huutokaupan rakentamisvaihetta, on ActiveBPEL erinomainen valinta työnkulkukoneeksi, sillä se tarjoaa täydellisen yhteensopivuuden mallinnusohjelman kanssa. Lisäksi ActiveBPEL Designer sisältää virtuaalisen työnkulkukoneen, jolla prosessia pystyy testaamaan jo mallinnusvaiheessa, aivan kuten tässä työssä tehtiin.

Toinen merkittävä ActiveBPEL-koneen etu on sen perustuminen avoimeen lähdekoodiin. Konetta kehitetään aktiivisesti ja se on vapaasti ladattavissa verkosta, ja periaatteessa kuka tahansa voi osallistua tuotteen kehittämiseen. Näin yrityksillä on mahdollisuus kehittää ohjelmistoa sähköisen huutokaupan tarpeet huomioon ottaen, jolloin siitä tulisi entistä kiinnostavampi, ja sitä kautta web-palveluihin perustuville sähköisille huutokaupoille olisi potentiaalia syntyä niiden vaatima kriittinen massa.

Omalta osaltaan ActiveBPEL:n houkuttelevuutta lisää helppokäyttöinen mallinnusohjelma, jolla prosessimalleja pystyy suunnittelemaan ilman vahvaa tietoteknistä taustaakin. Ohjelma tuottaa valmista BPEL-koodia, mutta valitettavasti käyttäjä ei pysty sitä käsin muokkaamaan. Kaikki muutokset on tehtävä ”graafisesti”, mikä hieman heikentää suunnittelun dynaamisuutta. Kaiken kaikkiaan ActiveBPEL vaikuttaa tämän tutkimuksen perusteella hyvältä vaihtoehdolta sähköisen huutokaupan työnkulkukoneeksi, mikäli huutokauppa toteutettaisiin kyseisellä menetelmällä.

Jos kysymystä laajennetaan hieman yleisemmälle tasolle työn otsikon suuntaan, eli miten työnkulkukone ylipäätään soveltuu sähköisen huutokauppajärjestelmän moottoriksi, niin asiaan saadaan lisäulottuvuuksia. Työssä käytettyjen tutkimusmenetelmien valossa web-palvelut ja niitä pyörittävä työnkulkukone tarjoaa valtavat mahdollisuudet aivan uudenlaisten ”älykkäiden” palvelujen luomiseen verkossa, jossa koneet voivat kommunikoida keskenään ja tehdä vaikkapa tarjouksia tuotteista automaattisesti ilman ihmisen puuttumista asiaan. Teknologia ja työkalut ovat

siis valmiita, mutta asiaa hankaloittaa ihmisten skeptisyys sekä uuden teknologian mukanaan tuomat haasteet liiketoiminnalle.

Suurimmaksi ongelmaksi työn aikana muodostui seuraava rivi WSDL-dokumentissa: <wsdl:part name="endpointReference" type="sref:ServiceRefType"/>. EndpointReferencen avulla päästään käsiksi web-palveluun käyttäen tiettyä protokollaa ja tallennusformaattia, eli se lisää palvelun dynaamisuutta konkreettisia web-palveluja käytettäessä. Koska työ rakentui abstraktien web-palvelujen ympärille, ei kyseistä aktiviteettia voinut käyttää. Myöskään ActiveBPEL Designer ei suostu käynnistämään simulointia, mikäli prosessissa esiintyy virheitä. Ongelma ratkesi tekemällä prosessista staattisen siltä osin, eikä mainitulla muutoksella ole vaikutusta prosessin toiminta-ajatukseen saati lopputulokseen.

Toinen hankaluus työssä johtui lähdemateriaalin niukkuudesta. Koska kyse on suhteellisen uudesta teknologiasta ja uudesta toteutustavasta B2B-markkinapaikoilla, ainoat lähdemateriaalit olivat käytännössä verkkojulkaisuja. Onneksi työn molemmat tarkastajat ovat työssä käsiteltyjen asioiden asiantuntijoita, ja heidän antamansa tuki ja opastus aiheeseen auttoi allekirjoittanutta merkittävällä tavalla.

7 YHTEENVETO

Erilaisia sähköisiä huutokauppajärjestelmiä on lukuisia, samoin kuin niissä käytettäviä teknologioita ja arkkitehtuuriratkaisuja. Tässä työssä esiteltiin web-palveluihin perustuva huutokauppa, josta työn kokeellisessa osuudessa toteutettiin yksi osa. Tällä periaatteella toteutettua täysimittaista sähköistä huutokauppaa ei vielä ole olemassa, ja tulosten hyödyntäminen liiketoiminnassa vaatii lisää tutkimusta aiheesta.

Työn kantavana ajatuksena oli tutkia työnkulkukoneen ja erityisesti ActiveBPEL-koneen soveltuvuutta web-palvelupohjaisen sähköisen huutokaupan moottorina. Tutkimus suoritettiin mallintamalla ja testaamalla kahden osapuolen, ostajan ja myyjän, tiedot tallentava rekisteröintiprosessi ActiveBPEL Designer -ohjelmalla. Sitä ennen prosessi suunniteltiin abstraktilla tasolla BPMN-notaation avulla.

Toteutettu rekisteröintiprosessi on itsekin web-palvelu, joka toimii koordinaattorina ja ohjaa toisia mukana olevia web-palveluja. Tällaista keskusprosessin sisältävää web-palvelujen yhteistyötä kutsutaan orkestroinniksi. Muut mukana olevat web-palvelut eivät tiedä mukanaolostaan prosessissa, eivätkä siitä, että ne ovat osallisena korkeamman tason liiketoimintaprosessissa. Orkestrointi mahdollistaa myös pitkäkestoiset useita sovelluksia ja yrityksiä koskevat prosessit, jollaisia huutokauppaprosessit selvästi olisivat.

Tutkimuksen tuloksena syntyivät ActiveBPEL Designer -ohjelmalla toteutettu prosessimalli, prosessimallista tuotettu BPEL-koodi, simuloinnin tuloksena syntynyt prosessiloki sekä WSDL-rajapintakuvaus. Vaikka niitä ei suoraan sellaisenaan voida käyttää huutokauppajärjestelmässä sidosten ja porttien puuttumisen vuoksi, on niiden periaatetta mukaillen mahdollista kehittää kokonainen järjestelmä. Tällöin keskusprosessilla olisi toisenlainen tarkoitus, ja tämän työn rekisteröintiprosessi olisi vain yksi web-palvelu muiden joukossa. ActiveBPEL-koneen vastuulla olisi hallita ja suorittaa BPEL-prosesseja, vastata tehtävien ajoittamisesta prosessissa sekä kutsua tarvittavia resursseja mukaan ajettavaan prosessiin. Huutokauppajärjestelmää

ylläpitävällä yrityksellä voi olla taustalla myös työkulun hallintajärjestelmä, jonka avulla monimutkaiset liiketoimintaprosessit ovat helpommin hallittavissa.

Sähköinen huutokauppa tuo monia uudistuksia perinteisiin huutokauppoihin verrattuna. Yksi merkittävimmistä eroista on ostajan ja myyjän ”roolien vaihtuminen”, jolloin ostajat ilmoittavat haluavansa ostaa jonkin tuotteen tai palvelun tietyin ehdoin, ja tavarantoimittajat tekevät tarjouksia sen perusteella. Hinta ei välttämättä ole ainoa ratkaiseva tekijä, vaan ostajayritys voi ratkaista huutokaupan voittajan parhaaksi katsomillaan preferensseillä. Uutta sähköisessä huutokaupassa on myös se, että siinä voi olla yhtä aikaa useita myyjiä ja ostajia, jolloin käynnissä on monia yksittäisiä transaktioita samanaikaisesti. Lisäksi tuotteita tai palveluja voi olla myytävänä kerralla useampia. Sähköiset huutokaupat mahdollistavat myös sellaisten tuotteiden ja ennen kaikkea palvelujen kauppaamisen, joita ei perinteisesti huutokaupoissa ole totuttu näkemään, kuten ohjelmointipalvelut. Liiketoiminnan kannalta sähköisen huutokaupan merkittävimpiä hyötyjä on kannattavuuden lisääntyminen alentuneiden transaktiokustannusten myötä.

Työssä esitelty yritysten välinen sähköinen huutokauppa soveltuu parhaiten tilanteisiin, joissa tarvitaan joustavuutta ja nopeaa reagointia markkinatilanteisiin. Käänteisen huutokaupan ansiosta ostajan on mahdollista löytää nopeasti paras mahdollinen tuote tai palvelu tietyn kriteerin mukaisesti (joka voi siis olla hinnan lisäksi vaikkapa toimitusaika tai kuljetusehdot). Esimerkkinä voidaan ajatella tilanne, jossa yrityksellä voisi olla varastosaldoa tarkkaileva ohjelma, joka tietyn tuotteen mennessä alle minimirajan automaattisesti ilmoittaisi tarvitsevansa kyseistä tuotetta 50 kappaletta kahden päivän kuluessa. Sen jälkeen toimittajat, joilla on tarjota ostajan vaatimusten mukaista tuotetta annetuin ehdoin, voisivat tehdä tarjouksia annetun aikarajan sisällä. Luonnollisesti toimittajalla on tällöin oltava johonkin julkiseen palvelurekisteriin (kuten UDDI) rekisteröity web-palvelu, josta se on löydettävissä. Huutokaupan päätteeksi tarjouskilvan voittanut toimittaja toimittaisi tuotteet ostajalle määräaikaan mennessä.

Vuoden 2000 alussa, kun B2B-markkinapaikat otettiin käyttöön, niille ennustettiin loistavaa tulevaisuutta. Kasvuvauhti ei kuitenkaan ole vastannut odotuksia, vaan niiden

käyttö on jopa vähentynyt. Syy tähän liittyy luottamukseen ja ihmisten asenteisiin uutta teknologiaa kohtaan. (Commission of the European communities, 2002) Asiasta on jo käynnissä tutkimuksia, ja kun yritysten ja ihmisten muutosvastarintaa saadaan hillittyä, voidaan sähköisille huutokaupoille taas odottaa valoisampaa tulevaisuutta niiden tuomien kustannussäästöjen kannustamina. Lisäksi web-palveluihin liittyvät teknologiat ovat alustariippumattomia, joten yrityksiltä ei vaadita suuria käyttöönottoinvestointeja, ja sitä kautta kynnys ottaa esimerkiksi sähköinen huutokauppa mukaan yrityksen liiketoimintaan on suhteellisen matala.

LÄHTEET

Aalst, van der. 2003. W.M.P, Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*. [verkkodokumentti]. Saatavissa: http://www.csd.uch.gr/~hy565/Papers/dont_go_with_flow.pdf

Aalto, A., Järvinen, V., Halonen, V., Wihuri, P., Juote, T. 2000. Sähköinen liiketoiminta. KHT -yhdistyksen palvelu, ISBN 951-8993-77-7

Active Endpoints, Inc. 2007. [verkkosivusto]. Saatavissa: http://www.active-endpoints.com/

Commission of the European communities. 2002. COMMISSION STAFF WORKING PAPER on B2B Internet trading platforms: Opportunities and barriers for SMEs - A first assessment. [verkkodokumentti]. Saatavissa: http://ec.europa.eu/enterprise/ict/policy/doc/sec2002-1217en.pdf

Hollingsworth, D. 1995. The Workflow Reference Model. Workflow Management Coalition. [verkkodokumentti]. Saatavissa: http://www.wfmc.org/standards/docs/tc003v11.pdf

Hollingsworth, D. 2004. The Workflow Reference Model: 10 Years On. Teoksessa: Workflow Handbook 2004. Future Strategies Inc. Lighthouse Point, FL, USA. [verkkodokumentti]. Saatavissa: http://www.wfmc.org

Juric, M. 2005. A Hands-on Introduction to BPEL. Oracle. [verkkodokumentti]. Saatavissa: http://www.oracle.com/technology/pub/articles/matjaz_bpel1.html

Kalakota, R., Robinson, M. 2004. e-business 2.0: roadmap for success, Addison-Wesley, ISBN 0-201-72165-1

Karjalainen, N. 2000. Sähköinen liiketoiminta: haaste strategialle. WSOY, ISBN 951-0-24334-5

Kendall, K. 2001. B2B Online Reverse Auctions: What's New? School of Business-Camden, Rutgers University. [verkkodokumentti]. Saatavissa: http://www.decisionsciences.org/decisionline/Vol32/32_4/32_4ecom.pdf

Laamanen, K. 2003. Johda liiketoimintaa prosessien verkkona - ideasta käytäntöön. Laatukeskus, ISBN 952-5136-16-7

Lysons, K., Farrington, B. 2006. Purchasing and Supply Chain Management. Financial Times / Prentice Hall, ISBN 978-0-273-69438-0

Michelson, B. 2005. Elemental links. [verkkodokumentti]. Saatavissa: http://elementallinks.typepad.com/bmichelson/2005/09/index.html

Peltz, C. 2003. web services orchestration: a review of emerging technologies, tools, and standards. Hewlett-Packard. [verkkodokumentti]. Saatavissa: http://devresource.hp.com/drc/technical_white_papers/WSOrch/WSOrchestration.pdf

Puustjärvi, J. 2006. The gains of ontologies in electronic auctions. Lappeenranta University of Technology.

Savolainen, T., Saaren-Seppälä, K., Savolainen, S. 1997. Liiketoimintaprosessien luova virtaviivaistaminen. MET, ISBN 951-817-669-8

Seppänen, V. 2002. Asiakas/palvelin -arkkitehtuuri ja tietoverkot. Jyväskylän yliopisto. [verkkodokumentti]. Saatavissa: http://www.mit.jyu.fi/wikstrom/opetus/itk115/luennot/itk115-14_17_4slides.pdf

Seppänen, V. 2007. B2B ja SOA. Kehittämismenetelmät ja arkkitehtuurit liiketoiminnassa. Jyväskylän yliopisto. [verkkodokumentti]. Saatavissa: http://www.cs.jyu.fi/el/tjtse54_07/b2b-soa.pdf

Strecker, S., Seifert, S. 2004. Electronic sourcing with multi-attribute auctions. University of Karlsruhe. [verkkodokumentti]. Saatavissa: http://csdl.computer.org/comp/proceedings/hicss/2004/2056/07/205670165b.pdf

Systä, T. 2007a. Web service composition. Tampereen teknillinen yliopisto. [verkkodokumentti]. Saatavissa: http://www.cs.tut.fi/kurssit/OHJ-5201/materiaali/9_WS_coordination.pdf

Systä, T. 2007b. Web Services Description Language (WSDL). Tampereen teknillinen yliopisto. [verkkodokumentti]. Saatavissa: http://www.cs.tut.fi/kurssit/OHJ-5201/materiaali/7_WSDL_UDDI.pdf

Van Der Merwe, A.P. 2002. Project management and business development: integrating strategy, structure, process and project. International Journal of Project Management, Vol. 20.

Vuolajärvi, K. 2006. Liiketoimintamuutoslähtöinen tietoteknisen ratkaisun suunnittelu. Nokia Oyj. [verkkodokumentti]. Saatavissa: http://www.cs.jyu.fi/el/tjtse54_06/Luennot/Luento_architecture_design_final.pdf

W3 Schools. 2007. Web Services Tutorial. [verkkodokumentti]. Saatavissa: http://www.w3schools.com/webservices/default.asp

WfMC Workflow Management Coalition. 1999. Terminology & Glossary, The Workflow Management Coalition Specification. [verkkodokumentti]. Saatavissa: http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

World Wide Web Consortium (W3C). 2004a. Latest SOAP versions. [verkkodokumentti]. Saatavissa: http://www.w3.org/TR/soap/

World Wide Web Consortium (W3C). 2004b. Web Services Architecture. [verkkodokumentti]. Saatavissa: http://www.w3.org/TR/ws-arch/

World Wide Web Consortium (W3C). 2001. Web Services Description Language (WSDL) 1.1. [verkkodokumentti]. Saatavissa: http://www.w3.org/TR/wsdl

Zwiers, J., de Vos, B. 2005. Guide to BPEL. [verkkodokumentti]. Saatavissa: http://www.radikalfx.com/bpel/usage.html

LIITE 1. WSDL-dokumentti (auctionServiceInterface.wsdl)

```
<wsdl:definitions
   targetNamespace="http://example.com/auction/wsdl/auctionService/"
   xmlns:bpel="http://docs.oasis-
open.org/wsbpel/2.0/process/executable"
   xmlns:vprop="http://docs.oasis-open.org/wsbpel/2.0/varprop"
   xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
   xmlns:tns="http://example.com/auction/wsdl/auctionService/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <wsdl:message name="sellerData">
      <wsdl:part name="creditCardNumber" type="xsd:string" />
      <wsdl:part name="shippingCosts" type="xsd:integer" />
      <wsdl:part name="auctionId" type="xsd:integer" />
   </wsdl:message>
   <wsdl:message name="sellerAnswerData">
      <wsdl:part name="thankYouText" type="xsd:string" />
   </wsdl:message>
   <wsdl:message name="buyerData">
      <wsdl:part name="creditCardNumber" type="xsd:string" />
      <wsdl:part name="phoneNumber" type="xsd:string" />
      <wsdl:part name="auctionId" type="xsd:integer" />
   </wsdl:message>
   <wsdl:message name="buyerAnswerData">
      <wsdl:part name="thankYouText" type="xsd:string" />
   </wsdl:message>
   <wsdl:message name="auctionData">
      <wsdl:part name="auctionId" type="xsd:integer" />
      <wsdl:part name="amount" type="xsd:integer" />
      <wsdl:part name="sellerCreditCardNumber" type="xsd:string" />
      <wsdl:part name="shippingCosts" type="xsd:integer" />
      <wsdl:part name="buyerCreditCardNumber" type="xsd:string" />
      <wsdl:part name="phoneNumber" type="xsd:string" />
   </wsdl:message>
   <wsdl:message name="auctionAnswerData">
      <wsdl:part name="registrationId" type="xsd:integer" />
      <wsdl:part name="auctionId" type="xsd:integer" />
   </wsdl:message>

   <wsdl:portType name="sellerPT">
      <wsdl:operation name="submit">
         <wsdl:input message="tns:sellerData" />
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:portType name="sellerAnswerPT">
      <wsdl:operation name="answer">
         <wsdl:input message="tns:sellerAnswerData" />
      </wsdl:operation>
   </wsdl:portType>
```

(jatkuu)

LIITE 1. Jatkoa

```
   <wsdl:portType name="buyerPT">
      <wsdl:operation name="submit">
         <wsdl:input message="tns:buyerData" />
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:portType name="buyerAnswerPT">
      <wsdl:operation name="answer">
         <wsdl:input message="tns:buyerAnswerData" />
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:portType name="auctionRegistrationPT">
      <wsdl:operation name="process">
         <wsdl:input message="tns:auctionData" />
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:portType name="auctionRegistrationAnswerPT">
      <wsdl:operation name="answer">
         <wsdl:input message="tns:auctionAnswerData" />
      </wsdl:operation>
   </wsdl:portType>

   <vprop:property name="auctionId" type="xsd:integer" />
   <vprop:propertyAlias propertyName="tns:auctionId"
      messageType="tns:sellerData" part="auctionId" />
   <vprop:propertyAlias propertyName="tns:auctionId"
      messageType="tns:buyerData" part="auctionId" />
   <vprop:propertyAlias propertyName="tns:auctionId"
      messageType="tns:auctionData" part="auctionId" />
   <vprop:propertyAlias propertyName="tns:auctionId"
      messageType="tns:auctionAnswerData" part="auctionId" />

   <plnk:partnerLinkType name="sellerAuctionHouseLT">
      <plnk:role name="auctionHouse" portType="tns:sellerPT" />
      <plnk:role name="seller" portType="tns:sellerAnswerPT" />
   </plnk:partnerLinkType>
   <plnk:partnerLinkType name="buyerAuctionHouseLT">
      <plnk:role name="auctionHouse" portType="tns:buyerPT" />
      <plnk:role name="buyer" portType="tns:buyerAnswerPT" />
   </plnk:partnerLinkType>
   <plnk:partnerLinkType
      name="auctionHouseAuctionRegistrationServiceLT">
      <plnk:role name="auctionRegistrationService"
         portType="tns:auctionRegistrationPT" />
      <plnk:role name="auctionHouse"
         portType="tns:auctionRegistrationAnswerPT" />
   </plnk:partnerLinkType>

<wsdl:service name="auctionServiceBP"/>

</wsdl:definitions>
```

LIITE 2. BPEL-dokumentti (auctionhouse.bpel)

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
BPEL Process Definition
Edited using ActiveBPEL(tm) Designer Version 4.0.0 (http://www.active-
endpoints.com)
-->
<bpel:process
xmlns:as="http://example.com/auction/wsdl/auctionService/"
xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="auctionhouse"
suppressJoinFailure="yes" targetNamespace="http://auctionhouse">
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
location="auctionServiceInterface.wsdl"
namespace="http://example.com/auction/wsdl/auctionService/"/>
   <bpel:partnerLinks>
      <bpel:partnerLink myRole="auctionHouse" name="seller"
partnerLinkType="as:sellerAuctionHouseLT" partnerRole="seller"/>
      <bpel:partnerLink myRole="auctionHouse" name="buyer"
partnerLinkType="as:buyerAuctionHouseLT" partnerRole="buyer"/>
      <bpel:partnerLink myRole="auctionHouse"
name="auctionRegistrationService"
partnerLinkType="as:auctionHouseAuctionRegistrationServiceLT"
partnerRole="auctionRegistrationService"/>
   </bpel:partnerLinks>
   <bpel:variables>
      <bpel:variable messageType="as:sellerData" name="sellerData"/>
      <bpel:variable messageType="as:sellerAnswerData"
name="sellerAnswerData"/>
      <bpel:variable messageType="as:buyerData" name="buyerData"/>
      <bpel:variable messageType="as:buyerAnswerData"
name="buyerAnswerData"/>
      <bpel:variable messageType="as:auctionData" name="auctionData"/>
      <bpel:variable messageType="as:auctionAnswerData"
name="auctionAnswerData"/>
   </bpel:variables>
   <bpel:correlationSets>
      <bpel:correlationSet name="auctionIdentification"
properties="as:auctionId"/>
   </bpel:correlationSets>
   <bpel:sequence>
      <bpel:flow>
         <bpel:receive createInstance="yes"
name="acceptSellerInformation" operation="submit" partnerLink="seller"
portType="as:sellerPT" variable="sellerData">
            <bpel:correlations>
               <bpel:correlation initiate="join"
set="auctionIdentification"/>
            </bpel:correlations>
         </bpel:receive>
```

(jatkuu)

LIITE 2. Jatkoa

```
         <bpel:receive createInstance="yes"
name="acceptBuyerInformation" operation="submit" partnerLink="buyer"
portType="as:buyerPT" variable="buyerData">
            <bpel:correlations>
               <bpel:correlation initiate="join"
set="auctionIdentification"/>
            </bpel:correlations>
         </bpel:receive>
      </bpel:flow>
      <bpel:assign>
         <bpel:copy>
            <bpel:from part="auctionId" variable="sellerData"/>
            <bpel:to part="auctionId" variable="auctionData"/>
         </bpel:copy>
         <bpel:copy>
            <bpel:from>1</bpel:from>
            <bpel:to part="amount" variable="auctionData"/>
         </bpel:copy>
         <bpel:copy>
            <bpel:from part="creditCardNumber" variable="buyerData"/>
            <bpel:to part="buyerCreditCardNumber"
variable="auctionData"/>
         </bpel:copy>
         <bpel:copy>
            <bpel:from part="phoneNumber" variable="buyerData"/>
            <bpel:to part="phoneNumber" variable="auctionData"/>
         </bpel:copy>
         <bpel:copy>
            <bpel:from part="creditCardNumber" variable="sellerData"/>
            <bpel:to part="sellerCreditCardNumber"
variable="auctionData"/>
         </bpel:copy>
         <bpel:copy>
            <bpel:from part="shippingCosts" variable="sellerData"/>
            <bpel:to part="shippingCosts" variable="auctionData"/>
         </bpel:copy>
      </bpel:assign>
      <bpel:invoke inputVariable="auctionData"
name="registerAuctionResults" operation="process"
partnerLink="auctionRegistrationService"
portType="as:auctionRegistrationPT"/>
      <bpel:receive name="receiveAuctionRegistrationInformation"
operation="answer" partnerLink="auctionRegistrationService"
portType="as:auctionRegistrationAnswerPT"
variable="auctionAnswerData">
         <bpel:correlations>
            <bpel:correlation initiate="no"
set="auctionIdentification"/>
         </bpel:correlations>
      </bpel:receive>
```

(jatkuu)

LIITE 2. Jatkoa

```
      <bpel:flow>
         <bpel:sequence>
            <bpel:assign>
               <bpel:copy>
                  <bpel:from>
                     <bpel:literal>Thank You!</bpel:literal>
                  </bpel:from>
                  <bpel:to>$sellerAnswerData.thankYouText</bpel:to>
               </bpel:copy>
            </bpel:assign>
            <bpel:invoke inputVariable="sellerAnswerData"
name="respondToSeller" operation="answer" partnerLink="seller"
portType="as:sellerAnswerPT"/>
         </bpel:sequence>
         <bpel:sequence>
            <bpel:assign>
               <bpel:copy>
                  <bpel:from>
                     <bpel:literal>Thank You!</bpel:literal>
                  </bpel:from>
                  <bpel:to>$buyerAnswerData.thankYouText</bpel:to>
               </bpel:copy>
            </bpel:assign>
            <bpel:invoke inputVariable="buyerAnswerData"
name="respondToBuyer" operation="answer" partnerLink="buyer"
portType="as:buyerAnswerPT"/>
         </bpel:sequence>
      </bpel:flow>
   </bpel:sequence>
</bpel:process>
```