Sebastian Löfhjelm

KOMPONENTIT JA VERKKOPALVELUT ELEKTRONISTEN LIIKETOIMINTAPROSESSIEN TOTEUTTAMISEN VÄLINEINÄ

Tietojärjestelmätieteen
pro gradu –tutkielma
13.06.2002

Tietojenkäsittelytieteiden laitos
Jyväskylän yliopisto
Jyväskylä

TIIVISTELMÄ

Löfhjelm, Carl Thomas Sebastian
Komponentit ja verkkopalvelut elektronisten liiketoimintaprosessien toteuttamisen välineinä / Sebastian Löfhjelm.
Jyväskylä: Jyväskylän yliopisto, 2002.
84 s.
Tietojärjestelmätieteen pro gradu –tutkielma

Tässä tutkielmassa tarkastellaan elektronisten liiketoimintaprosessien toteuttamista EJB-komponenttiteknologian ja verkkopalveluiden avulla. Tarkastelun pohjana toimii elektronisen liiketoiminnan kehityksen mukanaan tuoma dynaamisuuden ja monimutkaisuuden kasvu.

Tutkielma on luonteeltaan käsitteellisteoreettinen ja se perustuu elektronista liiketoimintaa ja komponentteja käsittelevään kirjallisuuteen. Lähteinä on lisäksi käytetty ajankohtaisia artikkeleita.

Tutkimuksessa havaittiin EJB-komponenttimallin tarjoavan monipuolisen tuen liiketoimintalogiikan ja liiketoimintaprosessien kehittämiselle. Sen todettiin sopivan hyvin organisaatioiden sisäisten järjestelmien kehittämiseen, mutta olevan puutteellinen organisaatioiden välisten liiketoimintaprosessien toteuttamisen kannalta. Verkkopalveluiden nähtiin puolestaan olevan tärkeä teknologia organisaation välisten prosessien integroinnissa ja dynaamisessa palveluiden hakemisessa. Verkkopalvelut tarjoavat lisäksi uuden abstraktiotason komponenttimalleille, mikä mahdollistaa erillisten järjestelmien käsittelemisen komponentteina. Verkkopalveluiden teknologiat ovat kuitenkin yhä varsin kypsymättömiä ja hajanaisia, mikä rajoittaa niiden laajamittaista käyttämistä.

AVAINSANAT: Elektroninen liiketoiminta, prosessi, komponentti, Enterprise JavaBeans, verkkopalvelu

ABSTRACT

Löfhjelm, Carl Thomas Sebastian
Developing electronic business processes with components and web services / Sebastian Löfhjelm.
Jyväskylä: University of Jyväskylä, 2002.
84 pages
Master's thesis for computer science and information systems

This thesis examines how electronic business processes can be developed by using the EJB component technology and web services. The development of electronic commerce and the growing requirement for more dynamic services and management of complexity forms the basis of the thesis.

The thesis is theoretical in nature and it is based on literature dealing with electronic commerce and components. References also include many articles that are relevant and current.

In the thesis it was found that the EJB component model offers a versatile tool for developing business logic and business processes. The component model was seen suitable for the development of intra-organizational information systems, but it lacked important features needed for inter-organizational business process development. Web-services were on the other hand seen as an important technology for integrating organizational processes and in the dynamic discovery of services. The web service – model offers a new abstraction layer for traditional component models, which enables systems to be viewed as external components. The standards defining the web service technology are however still quite immature, which limits its applicability.

KEYWORDS: Electronic commerce, process, component, Enterprise JavaBeans, web service

SISÄLLYSLUETTELO

1 JOHDANTO

Tässä luvussa tarkastellaan tutkimuksen taustaan liittyviä tekijöitä sekä tutkimusongelmaa ja tutkimuksen tavoitteita. Luku sisältää myös aiheen rajaukseen liittyviä perusteluita sekä selvityksen käytetystä tutkimusmenetelmästä. Luvun lopussa esitellään lyhyesti tutkielman rakenne.

1.1 Tausta

Elektroninen liiketoiminta on 2000-luvulla muuttunut tärkeäksi osaksi yritysten toimintastrategiaa. Vaikka käsitteeseen liittyy paljon odotuksia ja jopa osin liioiteltua optimismia, niin voidaan odottaa, että yhä suurempi osa organisaatioiden prosesseista ja palveluista tullaan siirtämään tietoverkkoihin. Tietoverkot ovat myös mahdollistaneet uusien liiketoimintamuotojen syntymisen, kun asiakkaille on tarjottu mahdollisuus suorittaa transaktioita verkon kautta. Julkisten tietoverkkojen käyttäminen liiketoiminnassa aiheuttaa kuitenkin monia ongelmia sovelluskehitykselle, jonka on sopeuduttava nopeutuneeseen kehityssykliin ja verkottumisen mukanaan tuomaan järjestelmien monimutkaistumiseen.

Järjestelmät edellyttävät uusia keinoja, joilla monimutkaisuutta voidaan hallita ja joiden avulla voidaan rakentaa tehokkaita ja skaalautuvia järjestelmiä. Sovelluskehityksessä ollaankin siirtymässä rakenteisesta ja olioperusteisesta sovelluskehityksestä vaiheeseen, jota kutsutaan komponenttiperusteiseksi sovelluskehitykseksi. Komponenttiteknologian tavoitteena on pyrkiä kuvaamaan järjestelmä komponenttien joukkona, joka tarjoaa palveluja tarkasti määriteltyjen rajapintojen kautta. Komponenttiteknologian etu perinteiseen sovelluskehitykseen verrattuna on, että sen avulla sovelluksen kehittäjälle voidaan tarjota valmiita palveluja ja näin ollen vapauttaa hänet teknisten ongelmien selvittämisestä liiketoimintaongelmien ratkaisemiseen. Komponenttiteknologia mahdollistaa myös sovelluslogiikan siirtämisen asiakkaan päätelaitteelta tehokkaisiin

palvelimiin ja palvelukuorman hajauttamisen useiden palvelinten kesken, mikä parantaa asiakkaiden saamaa palvelua.

Komponenttiteknologian avulla pyritään luomaan modulaarinen sisäinen järjestelmä, jota muut järjestelmän osat voivat kutsua ja jota on helppo laajentaa ja ylläpitää. Komponenttiteknologioita on kuitenkin olemassa monia ja ne eivät ole keskenään kovin yhteensopivia. Lisäksi komponenttien käyttäminen vaatii niiden kutsurajapintojen tietämistä ja sellaista tietoa ei yleensä ole saatavilla, koska sitä pidetään salaisena ja yrityksen sisäisenä tietona. Verkossa tapahtuva liiketoiminta vaatii myös sitä, ettei asiakkaan tarvitse tietää niistä teknisistä ratkaisuista ja järjestelmistä, joita tietyn palvelun käyttäminen vaatii.

Komponenttiteknologiaa laajempi käsite on verkkopalvelu (web service), joka koostuu useista eri sisäisistä järjestelmistä muodostaen hajautetun ja dynaamisen järjestelmän. Verkkopalvelu voidaan ymmärtää komponenttina, jossa palvelua kutsutaan julkisesti saatavilla olevan rajapinnan kautta. Siinä missä komponentteja käytetään organisaation sisäisissä sovelluksissa, niin verkkopalvelut mahdollistavat organisaation ulkopuolisten sovellusten käsittelemisen komponenttina. Verkkopalveluissa pyritään käyttämään standardiprotokollia yhteensopivuuden turvaamiseksi ja kutsurajapintojen yksinkertaistamiseksi. Näin voidaan luoda dynaaminen sovellusten verkko, jonka avulla asiakkaalle voidaan tarjota juuri sitä palvelua, joka parhaiten sopii hänen tarpeisiinsa. Asiakkaan käytettäviksi saadaan myös sellaiset palvelut, joista hän ei itse ole tietoinen ja joita voidaan automaattisesti luokitella esimerkiksi tietoverkoissa toimivien agenttien avulla.

1.2 Tutkimusongelma ja tutkimuksen tavoite

Tämän tutkielman tutkimusongelmana on selvittää kuinka Enterprise JavaBeans – komponenttimalli ja verkkopalvelut soveltuvat elektronisten liiketoimintaprosessien toteuttamiseen.

Tutkimuksen tavoitteena on osoittaa, että nykyiset komponenttiteknologiat eivät riitä elektronisen liiketoiminnan verkostotalouden vaiheen saavuttamiseksi. Tavoitteena on lisäksi osoittaa, että järjestelmien integroituminen edellyttää abstraktimman tason komponenttimallia.

1.3 Aiheen rajaus ja tutkimusmenetelmä

Tutkimuksen taustalta löytyy elektronisen liiketoiminnan kehitysvaiheiden tarkastelu, jota ovat tutkineet muun muassa Emmerich, Finkelstein ja Piccinelli (2001), Järvelä ym. (2000) ja Björn ym. (2000). Eri tasoilla toimivia komponentteja ovat tutkineet Herzum ja Sims (1999), jotka ovat kehittäneet komponenttien karkeustasojaottelun, sekä Szyperski (2001), joka on tutkinut komponenttien ja verkkopalveluiden vastaavuutta.

Komponenttien osalta tarkastellaan palvelimissa toimivaa EJB (Enterprise JavaBeans) – komponenttimallia. Yhden komponenttimallin tarkasteluun ollaan päädytty tutkielman rajaukseen liittyvistä syistä sekä mahdollisuuteen tarkastella tarkemmin kyseistä komponenttimallia ja sen ominaisuuksia. Verkkopalveluiden osalta tutkimus keskittyy olemassa olevien teknologisten standardien tarkasteluun sekä niiden välisten yhteyksien selvittämiseen. Sekä valittua komponenttiteknologiaa että verkkopalveluita tarkastellaan liiketoimintaprosessien toteuttamisen näkökulmasta. Tutkielmasta on kokonaan rajattu pois tietoturva.

Tutkimus on luonteeltaan käsitteellisteoreettinen ja siihen ei liity empiiristä osaa. Tutkimuksessa on käytetty lähteinä sekä elektronista liiketoimintaa käsittelevää kirjallisuutta että komponenttiteknologiaa käsittelevää teknisempää kirjallisuutta. Tärkeän osan lähteistä muodostavat myös useat ajankohtaiset artikkelit.

Tutkimuksessa saavutut merkittävimmät tulokset liittyvät elektronisen liiketoiminnan prosessin näkökulmasta tehtyyn tarkasteluun EJB-komponenttimallin ja verkkopalveluiden osalta. EJB-komponenttiteknologian todettiin kuitenkin rajoittavan elektronisen liiketoiminnan sovellusten käyttömahdollisuutta, koska

komponenttimalleja on olemassa useita, ne ovat pääosin epäyhteensopivia. Tämän lisäksi palvelut ovat yleensä suljettuja eikä niitä pysty hakemaan ja kutsumaan organisaation ulkopuolelta. Tutkimuksessa havaittiin, että komponenttikäsitettä laajentamalla ja abstrahoimalla voidaan komponentteina käsittää myös organisaation ulkopuolella tietoverkoissa toimivat muut järjestelmät. Standardien rajapintojen avulla näiden järjestelmien tarjoamia palveluita voidaan käyttää samalla periaatteella kuin perinteisiä komponentteja. Näitä verkkopalveluiksi kutsuttuja palveluita voidaan hyödyntää palveluiden dynaamisessa haussa ja kutsumisessa sekä heterogeenisten järjestelmien integroinnissa, jota tarvitaan esimerkiksi useiden eri organisaatioiden välisten prosessien toteuttamisessa. Verkkopalveluiden todettiin kuitenkin olevan vielä niin kehittymättömiä ja standardien niin hajanaisia, ettei niitä voida laajamittaisesti käyttää dynaamisten prosessien toteuttamiseen.

1.4 Tutkielman rakenne

Tutkielman sisältö rakentuu siten, että toisessa luvussa käsitellään elektronista liiketoimintaa ja liiketoimintaprosesseja yleisellä tasolla. Kolmannessa luvussa esitetään komponenttiteknologiaan liittyviä määritelmiä ja käsitteitä. Luku sisältää myös komponenttien karkeustasojaottelun, joka muodostaa pohjan tutkielman komponenttiosuudelle. Neljännessä luvussa tutkitaan Enterprise JavaBeans –komponenttiteknologiaa sekä sen käyttöä liiketoimintaprosessien toteuttamisessa. Viidennessä luvussa tarkastellaan verkkopalveluarkkitehtuuria ja liiketoimintaprosessien toteuttamista hajautetussa ympäristössä. Kuudennessa luvussa pohditaan tutkielmassa esiin nousseita asioita.

2 ELEKTRONINEN LIIKETOIMINTA

Tässä luvussa tarkastellaan elektronista liiketoimintaa yleisesti sekä esitetään tasoluokittelu, joka kuvaa elektronisen liiketoiminnan kehittymistä. Lisäksi luvussa tarkastellaan elektronisen liiketoiminnan sovellustyyppejä ja niiden organisaatioille aiheuttamia vaatimuksia. Lopuksi tutkitaan liiketoimintaprosessien roolia elektronisessa liiketoiminnassa.

2.1 Elektronisen liiketoiminnan tasot

Elektroninen liiketoiminta mielletään helposti 2000-luvun Internet-huuman tuotokseksi ja yhdistetään enemmän tai vähemmän menestyksellisiin verkossa toimiviin yrityksiin. Käsitteen ideologia juontaa juurensa kuitenkin paljon kauemmaksi menneisyyteen sekä organisaatioihin, joita ei välttämättä mieltäisi elektronista liiketoimintaa harjoittaviksi yrityksiksi. Esimerkiksi EDI:ä (Electronic Data Interchange) on käytetty jo vuosikymmenten ajan tiedon välittämiseksi teollisissa yrityksissä sekä rahoitus-laitoksissa (Amor 2000, 7). Elektroninen liiketoiminta ei siten ole mikään uusi keksintö, vaan se on vain saanut uudenlaisen luonteen siirtyessään suurten yritysten välisistä yksityisistä tietoverkoista julkiseen, lähes kaikkialta tavoitettavissa olevaan verkkoon.

Elektroninen liiketoiminta on käsitteenä moniselitteinen ja se on vuosien mittaan saanut uusia merkityksiä sekä tarkentavia ilmaisuja. Kalakota ja Whinston (1996, 1) käyttävät käsitteitä elektroninen liiketoiminta ja elektroninen kauppa lähes synonyymeinä. He määrittelevät termin (electronic commerce) moderniksi liiketoimintametodologiaksi, joka vastaa organisaatioiden, kauppiaiden sekä asiakkaiden tarpeisiin alhaisemmista kustannuksista ja nopeammista toimitusajoista. Tämän lisäksi tuotteita ja palveluita pitää voida ostaa ja myydä tietoverkkojen välityksellä. Whinstonin ja Kalakotan (1996, 1) määritelmän voidaankin katsoa kuvastavan lähinnä elektronista kaupankäyntiä, jossa tietoverkko nähdään pelkkänä markkinointi ja myyntikanavana.

Matsuda, Yokoyama ja Yoshida (2002, 214-215) kuvaavat verkoissa toimivien palveluiden kehittymistä kolmivaiheisella mallilla. Ensimmäiseen vaiheeseen kuuluu heidän mukaansa erillisinä tarjottavien palveluiden käyttäminen Internetin standarditeknologioilla, kuten HTTP:llä ja HTML:llä. Toinen vaihe perustuu olemassa olevien palveluiden staattiseen integroimiseen organisaatioiden välillä ja siihen kuuluu olennaisena osana organisaatioiden välinen tiedonsiirto. Kolmanteen vaiheeseen he lukevat kuuluvaksi käyttäjän tarpeisiin perustuvan dynaamisen palvelun haun ja käyttämisen. (Matsuda, Yokoyama ja Yoshida 2002, 214-215.)

Emmerich, Finkelstein ja Piccinelli (2001, 169-170) jakavat elektronisen liiketoiminnan kahteen vaiheeseen. Ensimmäinen koostuu pääasiassa web-sivuista, portaaleista ja kauppapaikoista. Siihen kuulu myös organisaatioiden verkottuminen sekä Internetin käytön yleistyminen. Toisessa vaiheessa Internet muodostuu osaksi liiketoiminnan ydininfrastruktuuria. Heidän mukaan tässä vaiheessa organisaatioiden pitää yrittää muuttaa prosessinsa, tietonsa ja varansa digitaaliseen muotoon, jotta ne voivat tarjota niitä tietoverkoissa. (Emmerich, Finkelstein ja Piccinelli 2001, 169-170.)

Edellistä kattavamman jaottelun esittävät Järvelä ym. (2000, 7-9), jossa tietoverkoissa tapahtuva liiketoiminta on jaoteltu kolmeen vaiheeseen: elektroniseen kauppaan, elektroniseen liiketoimintaan sekä elektroniseen verkostotalouteen. Vaiheet on tarkemmin esitetty TAULUKOSSA 1.

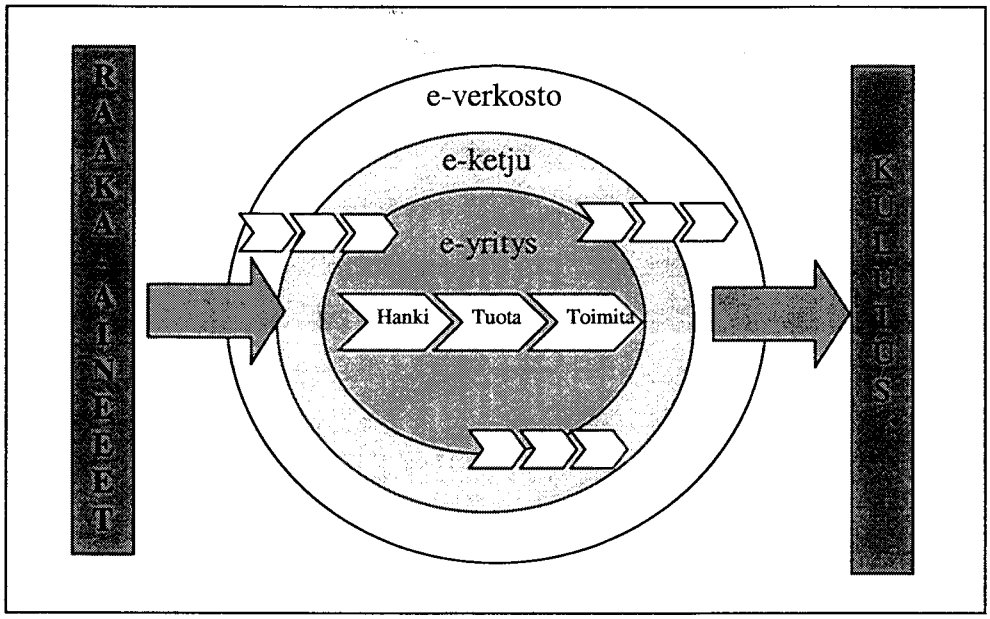
TAULUKKO 1. Elektronisen kaupan kehitysvaiheet Järvelä ym. (2000, 9).

	Elektroninen kauppa	**Elektroninen liiketoiminta**	**Elektroninen verkostotalous**
Prosessit	Erilliset e-kaupan prosessit	Integroidut ydinprosessit ja palveluprosessit	Osittain jaetut ja ulkoistetut
Teknologia	Kauppapaikka	Integroidut ERP-järjestelmät	Protokollat eli standardit ja platformit
Asiakassuhde	Erillinen	Pysyvä ja kiinteä	Asiakkaan ja/tai infomediaryn hallitsema

Heidän mukaansa *elektronisessa kaupankäynnissä* vain osa yrityksen prosesseista on muutettu tietoverkoissa toimiviksi. Tällaisia toimintoja voivat olla esimerkiksi tuotetietojen selaaminen ja tuotteen tilaaminen verkon kautta. *Elektronisen liiketoiminnan* vaiheessa yrityksen ydinprosessit on integroitu tietoverkkoihin ja asiakassuhdetta pyritään ylläpitämään muun muassa brandin ja asiakkuuden hallinnan avulla. Tämä vaihe voi myös sisältää yrityksen järjestelmien integroinnin toiminnanohjausjärjestelmiin. Kolmantena vaiheena Järvelä ym. (2000, 8) pitävät *elektronista verkostotaloutta*, jossa organisaation tietojärjestelmät on integroitu toisten organisaation järjestelmiin. Elektronisessa verkostotaloudessa yritykset voivat ulkoistaa prosessejaan toisten yritysten hoidettaviksi, mikä vaatii yhteisiä protokollia sekä standardeja. (Järvelä ym. 2000, 7-9.)

Elektronisen liiketoiminnan vaiheista on esittänyt mallin myös Björn ym. (2000, 52-55). He jakavat elektronisen liiketoiminnan viiteen eri vaiheeseen, jotka ovat: staattinen, vuorovaikutteinen, transaktiopohjainen, toimeenpaneva ja asiakassuhdetta hallitseva. *Staattisessa* vaiheessa yritykset välittävät verkossa tietoa itsestään ja toiminnastaan, mutta tieto on hyvin staattista ja yksisuuntaista. *Vuorovaikutteisessa* vaiheessa yritykset ilmoittavat verkossa jo liiketoimintaan liittyvää tietoa, kuten tuotteidensa hinta ja saatavuustietoja. Nämä tiedot vaativat kuitenkin manuaalista ylläpitoa, minkä vuoksi niiden ajantasaisuus voi olla heikko. *Transaktiopohjainen* vaihe mahdollistaa tilausten jättämisen, maksujen käsittelyn ja tilausten käsittelyn verkkopohjaisesti. Tällöin yritys on jo integroinut osan prosesseistaan tietoverkkoihin, mutta integrointi muihin liiketoimintaprosesseihin on edelleen heikko. *Toimeenpanevassa* vaiheessa yrityksen liiketoimintaprosesseja on integroitu tilaus-toimitus –ketjuun, johon kuuluvat toimittajat, liiketoimintakumppanit ja asiakkaat. Viidentenä vaiheena Björn ym. (2000, 54) näkevät *asiakassuhteen hallinnan* vaiheen, jossa tärkeällä sijalla ovat asiakkaiden ja liiketoimintakumppanien tietojen tehokas kerääminen ja hyödyntäminen.

Björn ym. (2000, 56) ovat edellä esiteltyjen elektronisen liiketoiminnan vaiheiden pohjalta luoneet digitaalisen tarjontaketjun johtamismallin. Malli on esitetty KUVIOSSA 1.



KUVIO 1. Digitaalisen tarjontaketjun johtamisen tasot Björnin ym. (2000, 56) mukaan.

Digitaalisen tarjontaketjun mallin sisimmällä kehällä sijaitsee e-yritys, jonka sisäiset prosessit on integroitu keskenään. Toimintaa ohjataan tavallisesti ERP (enterprise resource planning) –järjestelmän avulla. Seuraavalla kehällä sijaitsee e-ketju, jonka tarkoituksena on kuvata useamman kuin yhden organisaation välistä prosessi-integraatiota. Integraatio on perinteisesti suoritettu EDI:n avulla, mikä on edellyttänyt organisaatioiden välistä sopimusta tiedonsiirron periaatteista. Uloimmalla kehällä sijaitseva e-verkosto on tarjontaketjun viimeinen vaihe, jossa organisaatioiden väliset prosessit voivat integroitua toisiinsa mahdollisimman dynaamisesti kysyntä- ja tarjontatilanteen mukaan. (Björn ym. 2000, 56-58.)

Kaikissa edellä kuvatuissa malleissa on havaittavissa organisaation sisäisen ja ulkoisen integraation lisääntyminen elektronisen liiketoiminnan kehittyessä. Tämä aiheuttaa samalla sovellusten monimutkaistumista, koska niihin joudutaan lisäämään liiketoimintaprosessien sisältämää logiikkaa. Integraatiotason kasvu aiheuttaa lisäksi sovellusten välisten yhteyksien lisääntymistä, mikä edelleen lisää sovellusten

monimutkaisuutta. Tällöin myös sovellusten abstraktiotason pitäisi vastaavasti nousta, jotta monimutkaisuutta voidaan hallita.

2.2 Elektronisen liiketoiminnan sovellustyyppejä

Elektronisen liiketoiminnan sovelluksia voidaan luokitella usealla tavalla. Yleisimmin käytetty luokittelu on sovelluksen käyttäjäosapuolten mukaan tehty jako. Tällöin puhutaan yleensä organisaation ja kuluttajan välisestä (Business To Consumer, B2C) ja organisaatioiden välisestä (Business To Business, B2B) elektronisesta liiketoiminnasta. Vaikka molemmat sovellustyypit ovat elektronisen liiketoiminnan sovelluksia, niin niissä on havaittavissa tiettyjä ominaispiirteitä, jotka erottavat ne toisistaan.

2.2.1 Kuluttajille suunnatut sovellukset

Mayn (2000, 82-83) mukaan organisaation ja kuluttajien välinen elektroninen liiketoiminta keskittyy hänen mukaansa yksilön tekemiin transaktioihin ja siihen kuuluu yksilöille tarjottavat palvelut, kommunikointitavat ja henkilökohtaisten tarpeiden tyydytys.

Allamaraju ym. (2001, 18-19) puolestaan määrittelevät organisaation ja kuluttajien välisen elektronisen liiketoiminnan sovelluksen sellaiseksi sovellukseksi, joka tarjoaa organisaation rajapinnan kuluttajien käyttöön. Organisaatiot tarjoavat tällöin yleensä palveluitaan web-käyttöliittymän kautta. Näitä palveluita pyritään yleensä tarjoamaan kuluttajille personoidusti asiakasuskollisuuden lisäämiseksi. (Allamaraju ym. 2001, 18-19.)

Kuluttajille suunnattu elektroninen liiketoiminta on hyvin henkilökohtaista ja keskittyy ennen kaikkea kuluttajan ja palvelua tarjoavan organisaation väliseen kanssakäymiseen. Tämän vuoksi tärkeäksi tekijäksi muodostuu organisaation tietojärjestelmien kyky

palvella asiakkaita. Elektronisen liiketoiminnan sovelluksilla saattaa olla suuria käyttäjämääriä, jolloin järjestelmien sisäinen rakenne voi muodostua kriittiseksi tekijäksi niiden käytettävyyden kannalta. Kuluttajille suunnatut sovellukset voidaan mieltää edustajärjestelmäksi, joka on integroitu organisaation sisäiseen taustajärjestelmään.

2.2.2 Organisaatioiden väliset sovellukset

Mayn (2000, 85-86) mukaan organisaatioiden välinen elektroninen liiketoiminta on helpompaa kuin kuluttajille suunnattu, koska yritykset ovat tottuneet pitkiin liiketoimintasuhteisiin ja niillä on valmiit sopimukset, joita yleensä myös noudatetaan. Lisäksi elektronista liiketoimintaa harjoittavat yritykset tietävät tarkalleen kenen kanssa ne ovat liiketoimintasuhteessa ja pitkät sopimukset aikaansaavat sen, että transaktiot ovat yleensä säännöllisiä eikä toimittajaa vaihdeta helposti (poikkeuksen tästä muodostavat huutokaupat, joissa ostaja ja myyjä tekevät tarjouksia hakien parasta kauppakumppania). Organisaatioiden välisissä elektronisen liiketoiminnan sovelluksissa ei myöskään ole tarvetta tarkkailla asiakkaiden ostokäyttäytymistä siinä määrin kuin kuluttajille suunnatuissa sovelluksissa. EDI-pohjaisissa yksityisissä tietoverkoissa toimivat sovellukset ovat myös täysin eri luonteisia kuin esimerkiksi avoimessa Internetissä toimivat sovellukset. Edellisessä toimittajat ovat pitkälti teknisessä sidoksessa asiakkaaseensa, kun taas jälkimmäisessä pienten toimittajien neuvotteluvoima voi kasvaa, jos ne voivat vapaammin valita asiakkaansa. Tällaista kehitystä edesauttavat yhteiset standardit, joita kaikki voivat vapaasti käyttää. (May 2000, 85-86.)

Organisaatiot, joilla on useita eri liiketoimintakumppaneita, joutuvat yleensä käyttämään erilaisia protokollia eri kumppaneiden kanssa. Organisaatioilla voi myös olla lukuisia eri sisäisiä järjestelmiä, joihin kaikkiin on luotava erillinen integraatio. Integrointipisteiden lukumäärä kasvaa nopeasti yhteyksien määrän lisääntyessä, jolloin ohjelmallinen integrointi voi muodostua työlääksi. (Bussler 2002, 69-70.)

Organisaatioiden välisissä sovelluksissa tärkeitä tekijöitä ovat etenkin sovellusten yhtymäkohdat. Sovellusten luonteesta riippuu, kuinka tärkeä tekijä esimerkiksi uusien järjestelmäyhteyksien konfiguroinnin nopeus on. Organisaatioiden sisäisten sovellusten rajapinnat ovat lisäksi yleensä piilossa palomuurien takana, jolloin niiden tarjoamia palveluita ei pääse käyttämään ulkopuolelta. Vaikka palveluiden rajapinnat olisivatkin kutsuttavissa, niin ongelmaksi muodostuu oikeiden kutsurajapintojen löytäminen, koska niistä ei yleensä ole julkisesti saatavissa olevaa dokumentaatiota. Edellä mainittujen seikkojen vuoksi organisaatioiden välisten taustajärjestelmien integrointi vaatiikin yleensä hyvin paljon manuaalista työtä ja kohdeorganisaation järjestelmien tuntemista.

2.3 Liiketoimintaprosessi

Yksi elektronisen liiketoiminnan sovellusten keskeinen käsite on liiketoimintaprosessi. Seuraavaksi tarkastellaan erilaisia prosessin määritelmiä ja sitä mistä prosessi lopulta rakentuu. Hammer ja Champy (1993, 35) määrittelevät prosessin erilaisten tehtävien joukoksi, joka ottaa vastaan syötteitä ja tuottaa tulosteita. Yksittäiset tehtävät ovat tärkeitä prosessin toimivuuden kannalta, mutta asiakkaan kannalta tärkeää on vain lopputulos. (Hammer ja Champy 1993, 35.) Prosessi on tämän määritelmän mukaan yksittäisiä tehtäviä koordinoiva yksikkö, jonka tarkoituksena on varmistaa näiden tehtävien oikeanlainen suoritus.

Matena ja Stearns (2000, 18-19) määrittelevät liiketoimintaprosessin liiketoimintaolioksi, joka sisältää käyttäjän ja liiketoimintakohteiden välisen vuorovaikutuksen. Liiketoimintakohteet puolestaan ovat liiketoimintaoliota, jotka sisältävät organisaation kannalta relevanttia tietoa. Liiketoimintaprosessit muokkaavat liiketoimintakohteiden tilaa, joka yleensä on pysyvästi tallennettu tietokantaan. Myös liiketoimintaprosessilla itsellään voi olla tila. Tämä tilan elinkaari on yleensä sama kuin liiketoimintaprosessin elinkaari. (Matena ja Stearns 2000, 18-19.)

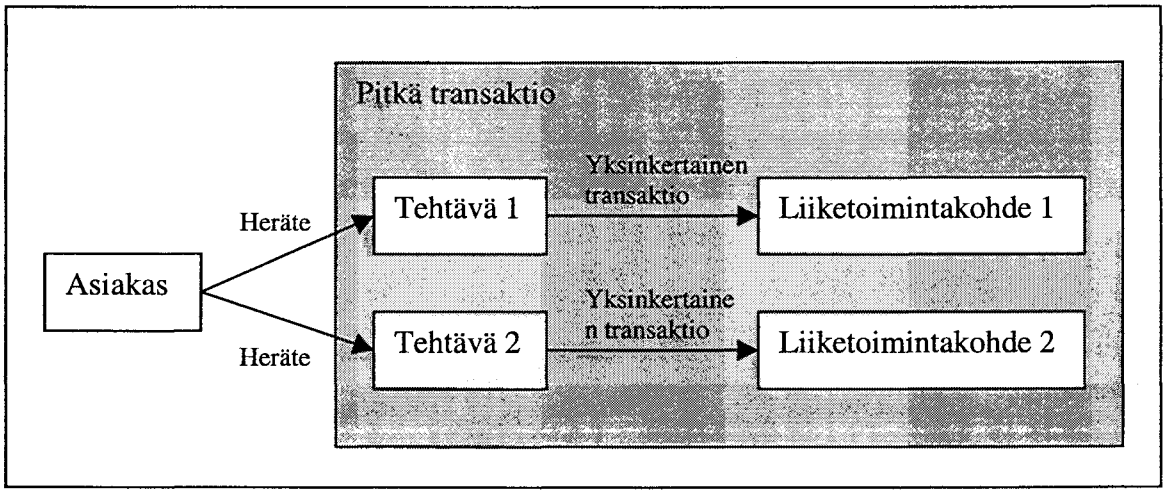
Matena ja Stearns (2000, 19) jakavat liiketoimintaprosessit kahteen tyyppiin: yhteistoiminnallisiin (collaborative) ja keskusteleviin (conversational).

Yhteistoiminnallisiin liiketoimintaprosesseihin kuuluu yleensä useita eri osapuolia ja vaiheita, jolloin myös itse liiketoimintaprosessista pitää tallentaa tilatietoa. Keskustelevissa liiketoimintaprosesseissa yksi käyttäjä kommunikoi järjestelmän kanssa, jolloin liiketoimintaprosessista itsestään ei välttämättä tarvitse tallentaa tilatietoa, jos ne ovat lyhytkestoisia tapahtumia. (Matena ja Stearns 2000, 19).

Daumin ja Schellerin (2000, 224-226) mukaan liiketoimintaprosessi koostuu tietyistä tehtävistä, jotka sijaitsevat liiketoimintaprosessin sisällä. Tehtävät ovat atomisia liiketoimintaan kuuluvia toimintoja, kuten laskun lähettäminen. Liiketoimintatehtävät käyttävät liiketoimintakohteita tietojen varastointiin sekä tiedon hakemiseen, jolloin liiketoimintakohteet mallintavat yleensä tietokantaa. Elektronisen liiketoiminnan sovelluksessa asiakas suorittaa liiketoimintatehtäviä lähettämällä pyyntöjä sovellukselle. Daum ja Scheller kutsuvat näitä pyyntöjä herättimiksi, koska ne käynnistävät tietyn tehtävän suorittamisen. (Daum ja Scheller 2000, 224-226.)

Liiketoimintaprosessiin kuuluu oleellisena osana myös transaktion käsite, jolla tarkoitetaan tiedon eheyden säilymistä liiketoimintaprosessin aikana. Prosessien kannalta on tärkeää, että tieto pysyy koko ajan eheässä tilassa, jotta toiset prosessit eivät saisi haltuunsa väärää tietoa. Transaktiot voidaan karkeasti jakaa kahteen päätyyppiin: yksinkertaiseen ja pitkään transaktioon. Yksinkertainen transaktio liittyy yhteen tehtävään prosessissa ja se voi koostua esimerkiksi tietokannan päivityksestä. Pitkä transaktio takaa sen, että tieto säilyy eheänä koko prosessin keston ajan, eli se varmistaa prosessiin kuuluvien atomisten tehtävien eheyden. (Daum ja Scheller 2000, 225-226.)

KUVIOSSA 2 on havainnollistettu liiketoimintaprosessin rakenne sekä siihen liittyvät elementit: heräte, tehtävä, liiketoimintakohde, yksinkertainen transaktio sekä pitkä transaktio.



KUVIO 2. Liiketoimintaprosessin malli Daumia ja Schelleriä (2000, 227) mukaillen.

2.3.1 Yksinkertainen transaktio

Liiketoimintaprosesseissa on mukana liiketoimintakohteita, joita prosessiin kuuluvat tehtävät käsittelevät. Liiketoimintakohteet sijaitsevat yleensä tietokannoissa, joissa ne ovat persistenssissä tilassa. (Daum ja Scheller 2000, 226.) Tiedon persistenssiys tarkoittaa sitä, että tieto pysyy tallessa vaikka sovellus sammutettaisiin tai tietoa käsittelevä järjestelmä välillä kaatuisi.

Tiedon eheys pyritään yleensä turvaamaan niin sanottujen ACID-määritysten (Atomicity, Consistency, Isolation and Durability) mukaisesti. *Atomisuus* tarkoittaa sitä, että transaktio joko toteutuu täydellisesti tai sitten ei ollenkaan. *Yhdenmukaisuudella* varmistetaan, että tieto siirtyy yhdenmukaisesta tilasta toiseen. *Eristettävyys* mahdollistaa usean samanaikaisen transaktion toteuttamisen toisistaan erillään. Tällöin voidaan varmistua siitä, että limittäin suoritettujen transaktioiden tulos on sama kuin peräkkäin suoritettujen transaktioiden. *Pysyvyydellä* voidaan varmistaa, että tieto on pysyvästi tallessa, jos tietoa muokannut transaktio on päättynyt normaalisti. (Gray ja Reuter 1993, 6.) Yksinkertaisilla transaktioilla tarkoitetaan tässä tutkielmassa sellaisia transaktioita, jotka ovat suhteellisen lyhytkestoisia, synkronisia ja toimivat yhden organisaation sisällä.

2.3.2 Pitkä transaktio

Liiketoimintaprosessien toteuttamiseen voi osallistua useita eri osapuolia ja ne saattavat kestää tunteja, päiviä tai jopa viikkoja, jolloin resurssien lukitseminen voisi pahimmillaan estää toisten prosessien toiminnan. Daum ja Scheller (2000, 225-226) määrittelevät tällaisen kokonaisen liiketoimintaprosessin kattavan transaktion pitkäksi transaktioksi. Pitkään transaktioon voi heidän mukaansa kuulua useita sessioita, käyttäjiä ja jopa toisia pitkiä transaktioita.

2.4 Yhteenveto

Luvussa määriteltiin elektronisen liiketoiminnan käsite sekä esiteltiin tarkemmin sen jakaantuminen erilaisiin tasoihin, joita olivat: elektroninen kauppa, elektroninen liiketoiminta sekä elektroninen verkostotalous. Elektronisessa kaupassa järjestelmien integrointi on vähäistä, kun taas elektronisessa liiketoiminnassa organisaation prosesseja on integroitu järjestelmiin. Verkostotaloudessa järjestelmät voivat kommunikoida ulkopuolisten järjestelmien kanssa, jolloin järjestelmät muodostavat verkoston. Integroinnin tason kasvu lisää kuitenkin samalla järjestelmien monimutkaisuutta.

Elektronisen liiketoiminnan sovellukset voidaan karkeasti jakaa kahteen osaa: kuluttajille suunnattuihin sekä organisaation välisiin sovelluksiin. Kuluttajille suunnatut sovellukset kuuluvat edustajärjestelmiin, koska ne ovat suoraan yhteydessä kuluttajaan. Organisaatioiden väliset sovellukset kuuluvat puolestaan taustajärjestelmiin ja niissä on yleensä useita integrointipisteitä.

Elektronisen liiketoiminnan yksi keskeinen käsite on liiketoimintaprosessi. Se voidaan jakaa pienempiin osiin, jolloin sen voidaan havaita koostuvan useasta pienemmästä tehtävästä. Tehtävät muokkaavat liiketoimintakohteiden tiloja ja niiden eheydestä pyritään varmistumaan transaktioiden avulla. Transaktiot voidaan jakaa karkeasti kahteen osaan: lyhyeen ja pitkään. Lyhytkestoiset transaktiot lukitsevat resurssit vain

lyhyeksi ajaksi, jolloin muiden prosessien ei tarvitse odottaa pitkään resurssien vapautumista. Lyhyet transaktiot soveltuvat etenkin edustajärjestelmien ja organisaation sisäisiin transaktioihin. Pitkiä transaktioita esiintyy organisaatioiden välisissä prosesseissa ja niiden ongelmana on resurssien pitkäaikainen lukitseminen.

3 KOMPONENTTITEKNOLOGIA

Tässä luvussa tarkastellaan erilaisia komponenttimalleja ja esitetään määritelmä komponenttiteknologiasta, minkä jälkeen tarkastellaan komponenttiteknologiaa tukevaa sovellusarkkitehtuuria. Luvussa osoitetaan myös niitä etuja ja haittoja, joita komponenttiperusteisen sovelluksen kehittämisessä pitää ottaa huomioon. Luvun lopussa tutkitaan lisäksi Herzumin ja Simsin (1999) esittämää komponenttien karkeustasojen luokittelua.

3.1 Komponenttiteknologia

Sovelluskehitystä on pitkään hallinnut olio-ohjelmoinnin ja sitä edeltäneen rakenteellisen ohjelmoinnin kausi. Näiden rinnalle on kuitenkin vahvasti tullut komponenttiteknologiaan perustuva sovelluskehitys, jossa sovellukset nähdään toisiinsa liittyneiden komponenttien joukkona. Seuraavaksi tarkastellaan lyhyesti erilaisia komponenttimalleja ja komponenttikäsitettä kirjallisuudessa esitettyjen määritelmien pohjalta.

3.1.1 Komponenttimallit

Komponentit tarvitsevat komponenttimallin, joka määrää sen kuinka komponentteja voidaan asentaa ympäristöön ja kuinka ne kommunikoivat keskenään. Komponenttimallit määrittelevät myös sen, kuinka komponentit voivat julkaista omat rajapintansa ja välittää tietoa. Komponenttimalleja on olemassa useita, mutta kolme tunnetuinta ovat:

- DCOM (Distributed Component Object Model)
- CORBA (Common Object Request Broker Architecture)
- EJB (Enterprise JavaBeans). (Hopkins 2000, 28.)

Microsoftin *DCOM* on kieliriippumaton komponenttimalli, joka on sidottu hyvin tiukasti kiinni Windows-maailmaan. Microsoftilla on oma sovelluspalvelin MTS (Microsoft Transaction Server), jossa komponentit voivat sijaita ja joka tarjoaa myös transaktiopalveluja. (May 2000, 205-206.)

CORBA on sekä kieli- että alustariippumaton komponenttimalli (Hopkins 2000, 28). Daumin ja Schellerin (2000, 120) mukaan sitä käytetäänkin paljon, kun halutaan yhdistää moderneja komponenttiperusteisia tietojärjestelmiä ja perintösovelluksia. CORBA-komponentit vaativat toimiakseen ajonaikaisen ympäristön, jota kutsutaan ORB:ksi (Object Request Broker). Ne voivat kommunikoida COM-komponenttien kanssa CORBA-DCOM –sillan avulla. (Daum ja Scheller 2000, 120.)

EJB on Java-kieleen sidottu palvelimissa toimiva komponenttimalli. Se etuna on käyttöjärjestelmäriippumattomuus, mikä tekee siitä siirrettävän eri käyttöjärjestelmien välillä. (Pour 2000). EJB-komponentit kykenevät kommunikoimaan myös muiden komponenttimallien kanssa (Daum ja Scheller 2000, 119). Cattellin ja Inscoren (2001, 19) mukaan EJB-komponenttimalli tarjoaa valmiina transaktio-, tietoturva- ja kuormanjakopalveluita, minkä johdosta sovelluskehittäjän ei tarvitse toteuttaa teknisiä palveluita, vaan hän voi keskittyä enemmän liiketoimintalogiikan toteuttamiseen.

3.1.2 Komponentti

Komponentti on ohjelmiston perusyksikkö komponenttiperusteisissa sovelluksissa. Sille on kuitenkin kirjallisuudessa olemassa useita erilaisia määritelmiä, minkä johdosta on tarpeen selvittää, mitä komponentilla tämän tutkielman yhteydessä tarkoitetaan. Herzumin ja Simsin (1999, 6-7) mukaan komponentti on ohjelmiston osa, joka on uudelleenkäytettävissä. Ohjelmistokomponentin pitäisi heidän mukaan sisältää ainakin hyvin määritellyt rajapinnat, joiden kautta komponenttia voidaan kutsua. Tämän lisäksi komponentti pitäisi voida asentaa järjestelmään muista komponenteista riippumatta. (Herzum ja Sims 1999, 6-7.)

Ambler, Jewell ja Roman (2001, 5) määrittelevät komponentin ohjelmistokoodiksi, joka toteuttaa hyvin määritellyt rajapinnat. Komponentit eivät heidän mukaansa itse toimi sovelluksina, vaan ne kootaan yhteen muiden komponenttien kanssa, jolloin saadaan aikaan toimiva sovellus. (Ambler, Jewell ja Roman 2001, 5.)

Myös Brownin (2000, 75) määritelmän mukaan komponentit ovat itsenäisiä toiminnallisia osia, jotka tarjoavat palveluita hyvin määriteltyjen rajapintojen kautta. Brown korostaa lisäksi, että rajapintojen tarkoituksena on eristää komponentin sisäinen toteutus ulkopuolisilta. (Brown 2000, 75.) Tämä tarkoittaa sitä, että komponenttia kutsuva asiakas ei ole sidottu komponentin toteutukseen vaan pelkästään sen rajapintaan. Rajapintojen käyttäminen mahdollistaa sellaisten järjestelmien kehittämisen, jotka ovat löyhästi toisiinsa sidottuja (Hopkins 2000, 27) ja toisaalta komponenttien toteutuksen korvaamisen toisilla komponenteilla (Roman 1999, 9).

Edellä esitetyt määritelmät komponentista ovat hyvin yleisiä ja sopivat myös käyttöliittymäkomponentteihin. Liiketoimintaprosessit vaativat kuitenkin muutamia lisäominaisuuksia komponentin käsitteeseen, jolloin puhutaan yritystason komponenteista. Herzumin ja Simsin (1999, 8) mukaan yritystason komponenteilla pitää olla verkon yli kutsuttava rajapinta, jotta niitä voidaan käyttää hajautetuissa sovelluksissa. Hajautetut sovellukset puolestaan aiheuttavat useita erityisvaatimuksia komponenteille, koska järjestelmien on kyettävä hallitsemaan tietoturva, samanaikaisuuden hallinta ja transaktiot. Tämän lisäksi komponenteilta vaaditaan suurta abstraktiotasoa, jotta järjestelmien ylläpito ei vaikeutuisi suuren komponenttimäärän vuoksi. Yhdeksi yritystason komponenttien erityispiirteeksi mainitaankin, että ne kuvaavat yleensä tiettyä liiketoimintakonseptia. (Herzum ja Sims 1999, 8.)

Brugalin, Fayadin ja Hamun (2000, 43) mukaan monet organisaatiot sijaitsevat maantieteellisesti hajallaan, jolloin sovellusten on pystyttävä kommunikoimaan toistensa kanssa sijainnista riippumatta. Komponenttiteknologia mahdollistaa komponenttien hajauttamisen erillisiin sovelluspalvelimiin, jolloin komponentit voivat pyytää toisiltaan palveluja tietoliikenneverkkojen yli. Komponenttia, jota voidaan

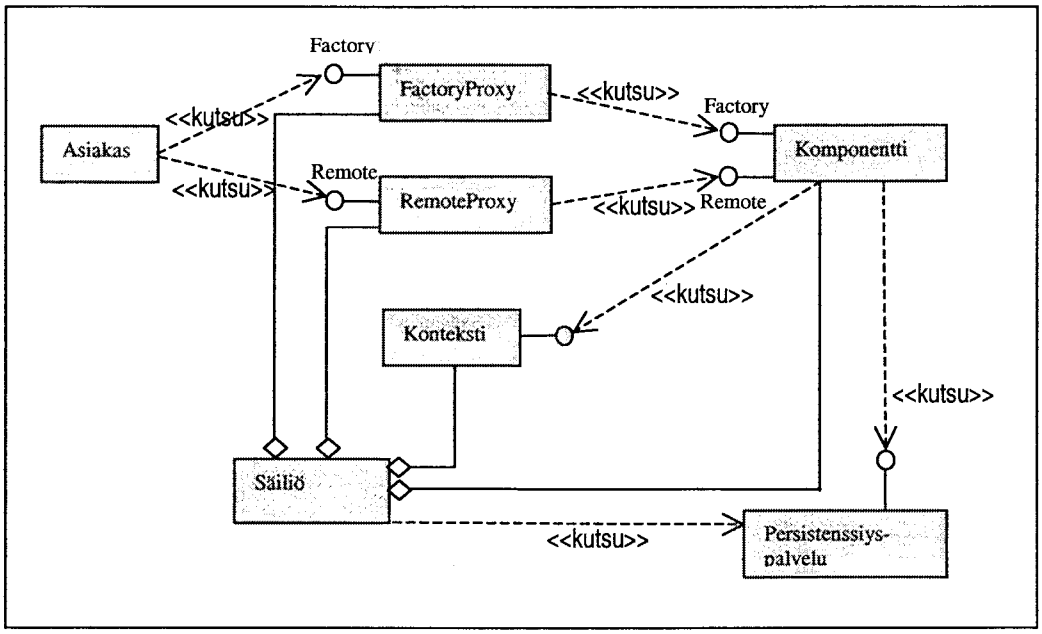
kutsua tietoliikenneverkkojen yli, kutsutaan hajautetuksi komponentiksi (Herzum ja Sims 1999, 39).

Herzum ja Sims (1999, 96) jakavat hajautetun komponentin kahteen osaan: funktionaaliseen ohjelmakoodiin ja erotuskerrokseen. Funktionaalisella ohjelmakoodilla tarkoitetaan komponentin sisältämää sovelluslogiikkaa. Erotuskerroksella puolestaan tarkoitetaan sitä osaa, joka erottaa komponentin funktionaalisen ohjelmakoodin sen ajonaikaisesta ympäristöstä. (Herzum ja Sims 1999, 96.)

Romanin (1999, 73; 77) mukaan EJB-komponenttimallissa funktionaalinen koodi sijaitsee EJB-komponentissa. Erotuskerros puolestaan on jaettu kahteen osaan: Home-olioon ja EJB-olioon. Asiakkaan kutsuessa komponenttia säiliö luo Home-olion avulla uuden EJB-olion, joka välittää varsinaisen pyynnön EJB-komponentille. (Roman 1999, 73; 77.)

Komponentit eivät siis suoraan kommunikoi keskenään, vaan kommunikointi tapahtuu komponenttien säiliön ja siinä olevien rajapintaluokkien kautta. Näiden luokkien välinen tietoliikenneprotokolla riippuu käytettävästä komponenttimallista. Esimerkiksi EJB-spesifikaatiossa 1.0 komponenttien välinen kommunikointiprotokolla on RMI (Remote Method Invocation). Myöhemmissä EJB-spesifikaatioissa on tarkoitus siirtyä käyttämään RMI/IIOP –protokollaa, joka mahdollistaa yhteensopivuuden CORBA:n kanssa. (Roman 1999, 77.)

EJB- ja COM-komponenttien arkkitehtuuri perustuu yleiseen malliin, jota kutsutaan yritystason komponentin sovelluskehykseksi. Malli kuvaa komponenttien yleisen rakenteen ja ajonaikaisen ympäristön. (Kobryn 2000, 35.) KUVIOSSA 3 on UML-notaatiota käyttäen kuvattu komponentin yleinen sovelluskehysmalli. Malli koostuu seitsemästä osasta, jotka ovat yhteydessä toisiinsa.



KUVIO 3. Yritystason komponentin sovelluskehysmalli (Kobryn 2000, 34).

KUVIOSSA 3 *asiakas* on mikä tahansa itsenäinen kokonaisuus, joka pyytää palvelua komponentilta. Asiakkaana voi olla esimerkiksi toinen komponentti. *FactoryProxy* on rajapinta, jonka kautta asiakas luo komponentin. Asiakas ei itse ole suoraan yhteydessä komponenttiin, vaan niiden välissä on *RemoteFactory*-rajapinta, joka kutsuu komponentin metodeja. Jokaisella komponentilla on *konteksti*, joka määrittelee komponentin transaktioiden tilat, persistenssiyden, tietoturvan ja sijoittelun. Komponentilla on myös käytössään *persistenssiyspalvelu*, joka mahdollistaa tietovarastoon tehtävät pyynnöt. Persistenssiyden hallinta on joko komponentin tai säiliön vastuulla. *Säiliö* tarjoaa komponentille ajonaikaisen ympäristön ja se koostuu komponentista, sen rajapinnoista ja kontekstista. (Kobryn 2000, 35-36.)

3.2 Komponenttiarkkitehtuuri

Komponentit eivät itsessään ole ajettavia ohjelmia, vaan ne vaativat toimiakseen tietyn toimintaympäristön, jota kutsutaan komponentin arkkitehtuuriksi. Komponenttien looginen arkkitehtuuri kuvaa komponenttien välisiä suhteita ja fyysinen arkkitehtuuri

kuvaa teknisiä asioita, kuten käytettäviä laitteita ja protokollia (Brown 2000, 87-88). Seuraavissa kappaleissa loogista arkkitehtuuria tarkastellaan sovellustasojen näkökulmasta ja fyysistä arkkitehtuuria sovelluspalvelimen ja säiliön näkökulmasta.

3.2.1 Looginen arkkitehtuuri

Looginen arkkitehtuuri jaetaan yleensä kolmeen eri pääluokkaan sen mukaan missä esityslogiikka, liiketoimintalogiikka ja tieto sijaitsevat. *Yksitasomallista* on May:n (2000, 187) mukaan kyse silloin, kun käyttäjä on yksin vastuussa sovelluksen asennuksesta, ylläpidosta sekä päivittämisestä. Yksitasomallissa sovelluksen käyttöliittymä, sovelluslogiikka ja tieto sijaitsevat yhdessä ja samassa koneessa (May 2000, 187). Yksitasomallin suurimpana ongelmana on se, että se ei mahdollista hajautettua tietojenkäsittelyä, joka muodostaa yhden modernien tietojärjestelmien kulmakivistä.

Asiakas-palvelin –mallissa asiakas tekee pyyntöjä, joihin palvelin pyrkii vastaamaan. Tässä mallissa on kaksi merkittävää eroa yksitasomalliin nähden. Ensinnäkin, asiakas ja palvelin voidaan hajauttaa eri tietojenkäsittelysolmuihin. Toiseksi asiakkaan ja palvelimen alustat ovat toisistaan riippumattomia edellyttäen, että niillä on yhteinen kommunikointiprotokolla sekä kommunikointiväylä. Asiakas-palvelin –mallista on useita variaatioita sen mukaan, kummassa päässä sovelluksen painopiste sijaitsee. Ääripäitä edustavat ruudunkaappaus, jossa asiakas pelkästään esittää graafisesti palvelimen tarjoamaa tietoa, ja toisaalta lihava asiakas, jossa osa toimintalogiikasta sijaitsee asiakkaan puolella. (May 2000, 186-188.)

Asiakas-palvelin –mallin suurimpana heikkoutena Adatia ym. (2001, 80-81) pitävät sitä, että se ei mahdollista liiketoimintalogiikan erottamista omaksi kokonaisuudekseen. Tämä olisi kuitenkin erityisen tärkeää sellaisissa sovelluksissa, joita pitäisi pystyä käyttämään erilaisten päätelaitteiden avulla. Asiakkaan ja palvelimen välinen tiukka kytkös on perimmäinen syy tämän mallin heikkouteen.

Monitasomallista on May:n (2000, 191) mukaan kyse silloin, kun sovelluksen tasot on jaettu vähintään kolmeen osaan siten, että ne sisältävät esityslogiikan, jaetun tietovaraston sekä näiden välistä yhteyttä ylläpitävän välitason. Tasoja voidaan jakaa edelleen pienempiin osiin, kun sovelluksessa havaitaan toiminnallisuutta, joka on eristettävissä omaksi tasokseen. (May 2000, 191.)

Mitä hienojakoisemmaksi tasot voidaan erotella, sitä modulaarisemmaksi sovellus muuttuu. Modulaarisuudella tarkoitetaan tässä sitä, että sovelluksen tiettyä toiminnallista osaa voidaan helposti mukauttaa uusiin vaatimuksiin. Elektronisen liiketoiminnan sovellusten yksi erityisvaatimus on muutoskyky, joka mahdollistaa sovelluksen nopean sopeuttamisen liiketoimintaprosesseissa tai –logiikassa tapahtuviin muutoksiin. Monitasoinen arkkitehtuuri, jossa sovelluksen osat on jaettu loogisesti yhtenäisiin kokonaisuuksiin helpottaa sovellusmuutoksia vaativien osien paikallistamisen. Sovelluksen jakaminen loogisiin osiin mahdollistaa myös komponenttiteknologian tehokkaan käytön, koska rajapintoja voidaan tällöin käyttää loogisten osien erottamiseen toisistaan.

Brown (2000, 53-54) toteaa tyypillisen välitason sisältävän yleensä web-palvelimen sekä sovelluspalvelimen. Web-palvelimen tehtävänä on vastaanottaa asiakkaan pyyntöjä ja palauttaa vastauksia asiakkaan päätelaiteelle sopivaan muotoon. Sovelluspalvelimen tehtävänä taas on toteuttaa liiketoimintalogiikka. (Brown 2000, 53-54.) Tämänkaltainen roolijako on sopivin komponenttiperusteisia sovelluksia ajatellen, koska komponentit voivat näin sijaita sekä loogisesti että fyysisesti erillään esitys- ja tietokerroksista, mikä helpottaa ylläpitoa ja luo järjestelmästä modulaarisen. Järjestelmän osien sijaitseminen eri tasoilla mahdollistaa myös horisontaalisen skaalautumisen, jolloin tietojenkäsittely-kapasiteettia voidaan lisätä uusien laitteiden avulla.

3.2.2 Fyysinen arkkitehtuuri

Komponenttien fyysiseen arkkitehtuuriin kuuluu sovelluspalvelin ja säiliö. Brownin (2000, 54) mukaan sovelluspalvelimen tehtävänä on liiketoimintalogiikan käsittely ja

tämän määritelmän mukaisesti sovelluspalvelimella tulisi sijaita sellaisia komponentteja, jotka toteuttavat liiketoimintalogiikkaa. Sovelluspalvelimen tehtävänä ei siis ole tiedon esittäminen eikä toisaalta tiedon tallentaminen.

Brown (2000, 55) mainitsee kuitenkin muutamia tehtäviä, joista sovelluspalvelin tavallisesti huolehtii. Ensinnäkin, suurissa liiketoimintaa tukevissa järjestelmissä tarvitaan yleensä transaktioiden hallintaa, joka voidaan antaa sovelluspalvelimen hoidettavaksi. Toiseksi, tietoturvan hallinta voidaan keskittää sovelluspalvelimeen, jolloin asiattomat eivät pääse käsiksi tärkeisiin tietoihin. Kolmantena tehtävänä Brown pitää tiedon hallintaa. Sovelluspalvelin tarjoaa yleensä rajapintoja näihin palveluihin, jolloin liiketoimintalogiikkaa toteuttavat komponentit voivat hyödyntää niitä. (Brown 2000, 55.)

Toinen tärkeä fyysiseen arkkitehtuuriin kuuluva osa on säiliö. Ambler, Jewell ja Roman (2001, 20) toteavat, että raja sovelluspalvelimen ja säiliön välillä on yleensä epäselvä, koska sovelluspalvelimen mukana tulee siihen kiinteästi liittyvä säiliö. Esimerkiksi EJB-spesifikaatio ei määrittele tarkkaa rajaa sovelluspalvelimen ja säiliön välillä, vaan ne käsitetään yleensä yhtenä kokonaisuutena. (Ambler, Jewell ja Roman 2001, 20.) Tässä tutkielmassa sovelluspalvelimesta puhutaan silloin, kun tarkoitetaan monitasoarkkitehtuurin välitasossa sijaitsevaa palvelinta. Sovelluspalvelin on näin ollen staattinen käsite. Säiliöstä puhutaan silloin, kun viitataan komponenteille tarjottaviin palveluihin sekä ajonaikaisiin ominaisuuksiin, jolloin säiliö nähdään dynaamisena kokonaisuutena.

Adatian ym. (2001, 10-11) mukaan EJB-komponenttimallissa säiliö on komponenttien ajonaikaisen ympäristön abstraktio. Se ei siten ole fyysisesti eristettävissä oleva osa, vaan pikemminkin tiettyihin sopimuksiin perustuva toiminnallinen määritelmä. Nämä *sopimukset* (component contract) määrittelevät komponentin elinkaaren hallinnan. Säiliöön kuuluu myös *palvelurajapintoja*, jotka sisältävät transaktion ja tietoturvan hallinnan, nimipalvelun, tietokanta-abstraktion sekä viestipalvelut. Kolmas säiliön tehtävä on niin kutsuttujen deklaratiivisten palveluiden hallinta. *Deklaratiivisuus* mahdollistaa komponenttien ominaisuuksien muuttamisen varsinaisen komponentin

ulkopuolelta ilman komponentin koodin muuttamista. Säiliön tehtäviin voidaan lisäksi lukea samanaikaisuuden hallinta, jota käytetään esimerkiksi säikeissä ja tietokantayhteyksissä. (Adatia ym. 2001, 10-11.)

3.3 Komponenttiteknologia sovelluskehityksessä

Miksi komponentteja pitäisi sitten käyttää sovelluskehityksessä? Komponenttien käyttämistä puoltavat monet seikat, jotka helpottavat sovellusten kehittämistä. Toisaalta komponentteihin liittyy myös lukuisia sovelluskehitystä haittaavia tekijöitä, jotka pitäisi ottaa huomioon komponenttiteknologian käyttämistä suunniteltaessa.

3.3.1 Komponenttien käyttämisen etuja

Sovellusten mukautumiskyky. Liiketoiminnassa tapahtuvien muutosten heijastaminen yrityksen tietojärjestelmiin voi nopeutua, jos käytetään komponenttiperusteista sovelluskehitystä. Muutosten tekeminen helpottuu, koska muutokset voidaan kohdistaa tiettyihin komponentteihin. (Herzum ja Sims 1999, 23.)

Uudelleenkäyttö. Yksi hyvin usein mainittu komponenttien käyttöä puoltava näkökulma on komponentteihin liittyvä uudelleenkäyttö. Uudelleenkäytöllä tarkoitetaan samojen komponenttien käyttämistä useissa eri sovelluksissa, jolloin komponentteja ei tarvitse kehittää joka kerta uudelleen, jos niiltä vaadittava toiminnallisuus on sama. Crnkovicin ja Larssonin (2000, 22) mukaan komponenttien uudelleenkäyttö hyödyttää sellaisia järjestelmiä, joilla on pitkä elinkaari. Resursseja kuluu enemmän tällaisten järjestelmien suunnitteluvaiheessa, mutta pidemmällä aikavälillä uudelleenkäytön hyödyt tulevat näkyviin. Komponenttien uudelleenkäyttöä voi esiintyä useilla tasoilla aina ohjelmakoodin kopioinnista kokonaisten järjestelmien uudelleenkäyttöön. (Crnkovic ja Larsson 2000, 22.)

Komponenttimarkkinat. Komponenttimarkkinoiden syntyminen on edellytyksenä tehokkaalle komponenttien uudelleenkäytölle ja komponenttiperusteiselle sovelluskehitykselle. Jotta komponenttimarkkinat voisivat syntyä, vaaditaan kuitenkin komponenttien tekijöitä, jotka tuottavat komponentteja myytäväksi. Kriittisen massan saavuttaminen edellyttää, että komponentit ovat monipuolisia ja toimittajia on useita. (Szyperski 1998, 6.) Komponenttimarkkinoiden etuna on, että organisaatioiden ei tarvitse itse kehittää monimutkaisia komponentteja, vaan että niitä voidaan ostaa edullisesti ja valmiiksi testattuina.

Komponenttimallien tarjoamat palvelut. Monet komponenttimallit helpottavat komponenttien kehittämistä tarjoamalla teknisiä palveluita valmiiksi rakennettuina, joiden toteuttaminen kestäisi liiketoimintalogiikan kehittäjiltä kauan ja vaatisi erikoisosaamista. Matena ja Stearns (2000, 9-11) mainitsevat muutamia seikkoja, jotka helpottavat EJB-komponenttien kehittämistä. EJB-komponenttimalli tarjoaa valmiina useita palveluita, kuten tietoturvan, transaktiot ja resurssialtaat. Tämän lisäksi komponenttimallissa on valmiiksi rakennettuna hajautettujen komponenttien käsittelylogiikka, jolloin sovelluksen kehittäjän ei tarvitse komponenttia kehittäessään ottaa kantaa siihen, missä tietojenkäsittelysolmussa se tulee lopulta sijaitsemaan. (Matena ja Stearns 2000, 9-11.)

3.3.2 Komponenttien käyttämisen ongelmia

Yleiskäyttöisten komponenttien tuottaminen. Komponenttien pitää olla riittävän yleiskäyttöisiä, jotta niitä voidaan käyttää erilaisissa sovelluksissa. Toisaalta liiallinen yleiskäyttöisyys tekee komponenteista monimutkaisia. Komponentin käyttäjän kannalta ideaalisinta olisi mahdollisimman pieni komponentti, joka toteuttaa juuri oikean tehtävän. Komponentin tuottajat puolestaan pyrkivät yleensä tuottamaan mahdollisimman vähän komponentteja, jotka ovat hyvin parametrisoitavissa. (Hopkins 2000, 30.) Näiden kahden näkemyksen välillä on nähtävissä ristiriita, joka saattaa hidastaa komponenttimarkkinoiden kasvua.

Puutteelliset mallintamismenetelmät. Vaikka UML tarjoaakin keinon mallintaa komponentteja, niin sen yleiskäyttöisyys voi vaikeuttaa mallintamista (Kobryn 2000, 33). Yleiskäyttöisyydellä Kobryn tarkoittaa sitä, että UML ei varsinaisesti sisällä notaatiota eri komponenttimalleille, vaan UML:n soveltaminen jätetään mallintajan vastuulle.

Komponenttien alustasidonnaisuus. Komponenttiperusteiset sovellukset tarvitsevat alleen komponenttiarkkitehtuurin, jossa ne voivat toimia. Jos tietojärjestelmä ei ole pohjimmiltaan komponenttiperusteinen, niin komponenttien käyttäminen ei sanottavasti auta järjestelmien muuttamisessa ja ylläpidossa. (Hopkins 2000, 30.)

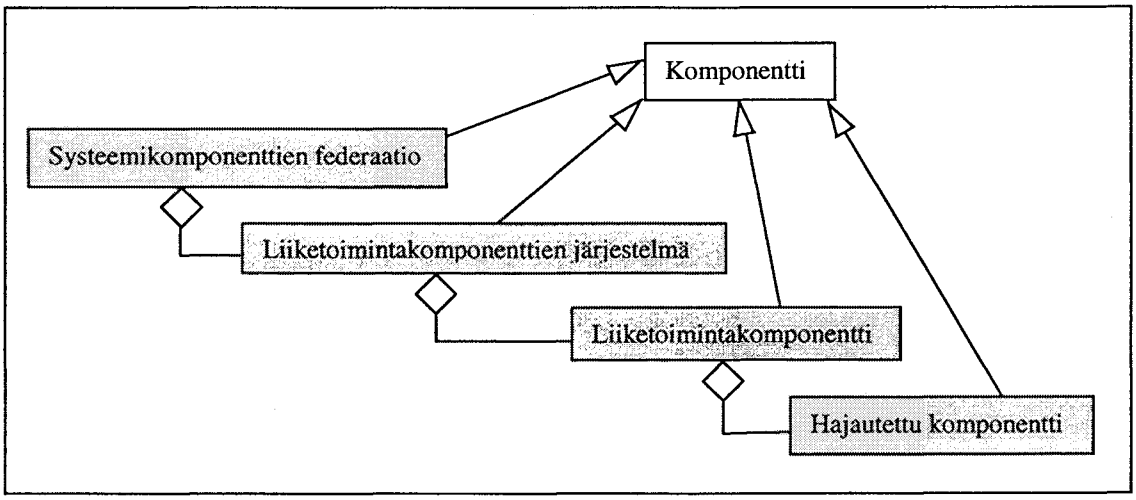
Komponenttien uudelleenkäyttö. Komponenttien uudelleenkäyttö on yksi komponenttien käyttämistä puoltava tekijä, mutta se on samalla myös yksi haittatekijä. Ranin (1999) mukaan komponenttien uudelleenkäyttö on hyvin vaikeaa, koska komponentit ovat harvoin täysin eristettävissä ympäröivistä komponenteista. Tämä johtuu siitä, että komponentit eivät ole yksinkertaisia, vaan pikemminkin monimutkaisia rakennelmia, jotka ovat ympäristöönsä sidoksissa tietoturvan, virheidenkäsittelyn, tilanhallinnan ja kommunikointi-infrastruktuurin kautta. (Ran 1999, 164.)

3.4 Komponenttien karkeustaso

Herzum ja Sims (1999, 36-39) jakavat komponentit neljään eri luokkaan niiden karkeustasoon perustuen. Tasot hienojakoisimmasta karkeimpaan ovat: hajautettu komponentti, liiketoimintakomponentti, liiketoimintakomponenttien järjestelmä ja systeemikomponenttien federaatio. (Herzum ja Sims 1999, 36-39.)

KUVIOSSA 4 on kuvattu eri komponenttitasojen suhde toisiinsa. Huomionarvoista kuviossa on se, että kaikki komponenttitasot periytyvät komponentista, minkä johdosta niillä on tiettyjä komponenteille tyypillisiä ominaisuuksia. Toiseksi, karkeajakoisempi komponentti koostuu aina hienojakoisemmista komponenteista, jolloin komponentit

muodostavat hierarkkisen rakenteen. Seuraavissa kappaleissa käsitellään tarkemmin jokaista neljää komponenttiluokkaa sekä vertaillaan niitä toisiinsa.



KUVIO 4. Komponenttien karkeustaso Herzumia ja Simsiä (1999, 37) mukaillen.

3.4.1 Hajautettu komponentti

Hajautettu komponentti on hienorakenteisin komponentin muoto ja se toteutetaan yleensä jollakin komponenttimallilla, kuten CORBA:lla, DCOM:lla tai EJB:llä. Sillä on muutama hajautetulle komponentille kuuluva ominaispiirre: Sen toteutus on erotettu kutsurajapinnasta, se voidaan laittaa ajoympäristöönsä muista komponenteista riippumatta ja sitä voidaan kutsua tietoverkon yli. (Herzum ja Sims 1999, 39.)

Herzumin ja Simsin (1999, 39) mukaan hajautettu komponentti on teknisin kaikista komponenteista ja sen toteuttamisesta vastaa funktionaalinen suunnittelija. Funktionaalisella suunnittelijalla he tarkoittavat sellaista komponentin kehittäjää, jonka ei tarvitse huolehtia komponentin toimintaan liittyvistä teknisistä yksityiskohdista, joita ovat muun muassa hajautettujen komponenttien paikantaminen, tiedon siirto komponenttien välillä, transaktiorajojen määritteleminen ja säikeistys. (Herzum ja Sims 1999, 39.)

Perinteisessä sovelluskehityksessä liiketoimintaoliot muuttuvat kehittämisvaiheiden rajapinnoissa ja niistä häviää tietoa viimeistään toteutusvaiheessa, kun ne käännetään moduuleiksi. Käännetyillä moduuleilla ei kuitenkaan enää välttämättä ole mitään tekemistä alkuperäisen liiketoimintakäsitteen kanssa. Hajautetun komponentin eräs merkittävä ominaisuus onkin jäljitettävyys, mikä tarkoittaa sitä, että komponentin käsite säilyy samana koko kehitysprosessin ajan aina analyysistä fyysiseen toteutukseen saakka. (Herzum ja Sims 1999, 80-81.)

3.4.2 Liiketoimintakomponentti

Hajautettua komponenttia laajempi käsite on liiketoimintakomponentti, joka toteuttaa yksittäisen liiketoimintakäsitteen tai –prosessin. Siihen kuuluvat kaikki ohjelmiston elementit, jotka ovat välttämättömiä itsenäisen ja uudelleenkäytettävän hajautetun järjestelmän osan toteuttamiseksi (Herzum ja Sims 1999, 40).

Herzumin ja Simsin (1999, 41) määritelmän mukaan liiketoimintakomponentti kuvaa komponenttia abstraktimmalla tasolla kuin hajautettu komponentti. Esimerkkinä he mainitsevat asiakkaan, joka on selvästi liiketoiminnallinen käsite, ja toisaalta päivämäärän, jota ei yksinään voida pitää liiketoiminnallisena käsitteenä. Liiketoimintakomponentit tarjoavat palveluita tekemällä yhteistyötä ja kommunikoimalla toisten liiketoimintakomponenttien kanssa. (Herzum ja Sims 1999, 41.)

Monitasoiset sovellusarkkitehtuurit mahdollistavat komponenttien sijoittelemisen eri tasoille niiden käyttötarkoituksen mukaan. Herzum ja Sims (1999, 41-43) toteavat, että liiketoimintakomponentti läpäisee käsitteenä kaikki arkkitehtuuriset tasot. Esimerkiksi asiakaskomponentti voi sijaita tietokannassa, sitä käsittelevässä liiketoimintakerroksen komponentissa ja lopulta käyttöliittymässä. Tämän lisäksi liiketoimintakomponentin käsitteeseen kuuluu analyysi-, toteutus- ja sijoitteluvaiheen tuotokset, minkä voidaan nähdä kuvastavan liiketoimintakäsitteen ajallista muotoutumista järjestelmän osaksi. (Herzum ja Sims 1999, 41-43.)

3.4.3 Liiketoimintakomponenttien järjestelmä

Liiketoimintakomponentteja yhdistämällä saadaan aikaan toimiva sovellus, jota kutsutaan liiketoimintakomponenttien järjestelmäksi. Sovellus käyttää hyväkseen liiketoimintakomponenttien palveluja ja ratkaisee jonkin liiketoimintaongelman, kuten laskujen käsittelyn. (Herzum ja Sims 1999, 43-44.)

Komponenttiperusteisen sovelluksen etu tavalliseen sovellukseen verrattuna on, että liiketoimintakomponentti voidaan tarpeen vaatiessa vaihtaa toiseksi ilman, että sovellusta tarvitsee linkittää uudestaan. Komponenttijärjestelmä voidaan siten koostaa olemassa olevista liiketoimintakomponenteista, jotka voivat olla joko organisaation sisäisiä tai ulkopuolelta ostettuja. Tämänkaltaista järjestelmän rakentamista kutsutaan valkoiseksi laatikoksi (white box assembly), mikä tarkoittaa sitä, että järjestelmän kokoaja näkee komponentit, joista järjestelmä koostuu. (Herzum ja Sims 1999, 45-46.)

3.4.4 Systeemikomponenttien federaatio

Herzum ja Sims (1999, 47) määrittelevät systeemikomponenttien federaation joukoksi toisiinsa liittyneitä liiketoimintakomponenttien järjestelmiä, joiden tehtävänä on auttaa tiedon prosessoinnissa loppukäyttäjiä, jotka saattavat sijaita eri organisaatioissa. Laajoja järjestelmiä ei Herzumin ja Simsin (1999, 194) mukaan enää rakenneta täysin alusta, vaan ne kootaan yhteen olemassa olevista järjestelmistä. Järjestelmiä joudutaan kokoamaan erillisistä järjestelmistä, koska yritykset fuusioituvat tai ne voivat muodostaa virtuaalisia organisaatioita (Herzum ja Sims 1999, 195). Eri järjestelmiä yhdistettäessä on tärkeää löytää yhteinen kieli, jolla järjestelmät voivat kommunikoida keskenään. Yhden vaihtoehdon tähän tarjoaa XML (Extensible Markup Language), jolla voidaan kuvata tietoa sisältävän dokumentin rakenne (Daum ja Scheller 2000, 143).

Herzumin ja Simsin (1999, 46) mukaan systeemitasolla tapahtuva järjestelmien integrointi tapahtuu musta laatikko –periaatteella (black box assembly), jolloin järjestelmän kokoaja ei pysty vaikuttamaan integroitavan järjestelmän sisäiseen rakenteeseen, mikä liiketoimintakomponenttien järjestelmän kohdalla on mahdollista.

3.5 Yhteenveto

Komponentit ovat ohjelmakoodia sisältäviä sovelluksen osia, jotka toimivat tietyssä ympäristössä. Komponenttien kutsurajapinta on erotettu itse komponentista, jotta komponentissa tehtävät muutokset tai komponentin vaihtaminen toiseen, eivät näkyisi asiakkaalle. Hajautetut komponentit voivat sijaita eri paikoissa ympäri verkkoa, mutta asiakkaan kannalta komponenttien sijainnilla ei ole merkitystä, koska komponentit ja niiden ympäristö huolehtivat komponenttien paikantamisesta.

Komponentit tarvitsevat toimiakseen jonkin komponenttimallin, joka tarjoaa komponentille ajonaikaisen ympäristön sekä määrittelee sen kuinka komponentit kommunikoivat ja kuinka rajapinnat muodostetaan. Yleisesti käytettyjä komponenttimalleja ovat CORBA, DCOM ja EJB.

Komponenttiarkkitehtuuri muodostaa komponentin ympäristön ja se voidaan jakaa loogiseen ja fyysiseen arkkitehtuuriin. Loogisella arkkitehtuurilla tarkoitetaan komponenttien suhdetta toisiinsa sekä niiden sijaintia eri sovellustasoilla. Loogisesta arkkitehtuurista on olemassa kolme erilaista mallia: yksitaso-, asiakas-palvelin- ja monitasomalli. Näistä monitasomalli tukee parhaiten komponenttien ominaisuuksia, koska se mahdollistaa sovelluksen hajauttamisen ja jakamisen komponentteihin erilaisten toiminnallisten tasojen mukaan.

Komponenttien fyysinen arkkitehtuuri puolestaan käsittää sovelluspalvelimen ja sen sisältämän säiliön. Sovelluspalvelin voidaan nähdä staattisena objektina, kun taas säiliö on enemmän ajonaikainen objekti. Komponentit toteuttavat yleensä säiliön tarjoaman sopimuksen, joka määrittelee ne palvelut, joita komponentti voi käyttää. Sopimus

mahdollistaa monien teknisten palveluiden käyttämisen, ilman että niitä tarvitsisi itse toteuttaa.

Komponenttien käyttämisellä voidaan saavuttaa monia etuja, kuten komponenttitason uudelleenkäyttö, komponenttimallin tarjoamien teknisten palveluiden hyödyntäminen ja sovellusten muokattavuuden helpottuminen. Komponenttiteknologialla on myös kääntöpuolensa, joka voi johtua yleiskäyttöisten komponenttien kehittämisen vaikeudesta, komponenttien monimutkaisuudesta aiheutuvasta uudelleenkäytön ongelmasta tai komponenttiperusteisen arkkitehtuurin puuttumisesta.

Komponentit voidaan jakaa neljään eri luokkaan niiden rakeisuuden mukaan. Hienojakoisimpaan luokkaan kuuluu hajautettu komponentti, joka voidaan nähdä kaikkien komponenttien pienimpänä osana. Edellistä karkeajakoisempaan luokkaan kuuluu liiketoimintakomponentti, joka toteuttaa yksittäisen liiketoimintaprosessin. Se voi koostua useammasta hajautetusta komponentista ja se voi kommunikoida muiden liiketoimintakomponenttien kanssa toteuttaakseen asiakkaan pyynnön. Liiketoiminta-komponenttien järjestelmä kuuluu kolmanteen karkeusluokkaan ja se voi koostua useista liiketoimintakomponenteista. Liiketoimintakomponenttien järjestelmä on itsenäinen toimiva sovellus, joka on rakennettu tietyn ongelman ratkaisemiseksi. Komponenttien federaatio on laajin komponentin käsite ja se koostuu useista erillisistä järjestelmistä. Järjestelmät kommunikoivat toistensa kanssa toteuttaakseen liiketoimintaprosesseja.

4 ENTERPRISE JAVABEANS KOMPONENTTIMALLI

Tässä luvussa tarkastellaan sitä, kuinka organisaation sisäisiä liiketoimintaa tukevia sovelluksia voidaan rakentaa käyttämällä J2EE:hen kuuluvaa EJB-komponenttiteknologiaa. EJB:n osalta tutkitaan erityisesti EJB 2.0-spesifikaation mukanaan tuomia uusia ominaisuuksia, jotka vaikuttavat muun muassa komponenttien tehokkuuteen.

Luvun alussa tarkastellaan kuinka liiketoimintaprosesseja voidaan mallintaa istuntokomponenttien avulla. Tämän jälkeen tarkastellaan kohdekomponenttia sekä sen suhdetta liiketoimintaprosessissa välitettävään tietoon. Luvun lopussa tutkitaan vielä viestipohjaista komponenttia sekä siihen liittyvää asynkronista viestinvälitystä.

4.1 Liiketoimintaprosessien mallintaminen

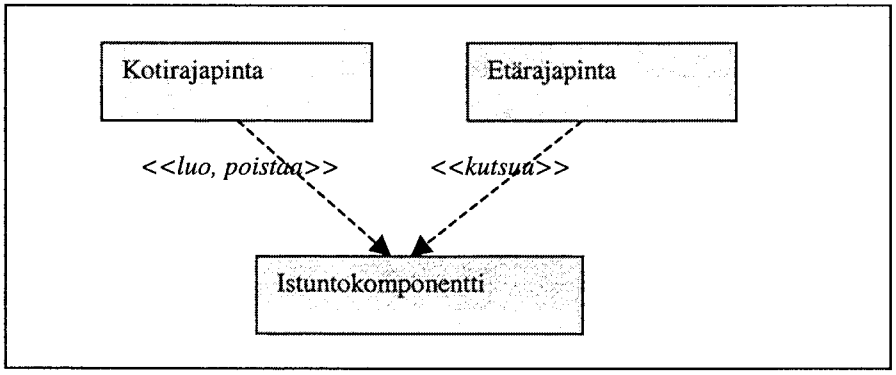
Yksi elektronisen liiketoiminnan sovelluksen keskeisiä piirteitä on, että se toteuttaa reaalimaailman liiketoimintaprosesseja. Liiketoimintaprosessien toteuttaminen EJB:llä edellyttää sellaisen komponenttityypin käyttämistä, joka parhaiten tukee prosessien hallinnointia ja käsittelyä. Seuraavaksi tarkastellaan tarkemmin liiketoimintaprosessin mallintamista istuntokomponentin avulla.

4.1.1 Istuntokomponentin rakenne

Istuntokomponentteja (Session Bean) käytetään EJB-komponenttimallissa kuvaamaan liiketoimintaprosesseja. Ne ovat nimensä mukaisesti istuntokohtaisia ja niiden elinkaari kestää yhtä kauan kuin komponenttia käyttävän asiakkaan istunto (Adatia ym. 2001, 23).

Istuntokomponentit jaotellaan niiden tarjoaman istunnon pituuden mukaan kahteen luokkaan: tilattomiin ja tilallisiin. Tilattomat komponentit eivät säilytä tietoa edellisistä istunnoista, joten niitä ei voida käyttää asiakaskohtaisen tiedon säilyttämiseen. Tilallinen istuntokomponentti säilyttää tilatietoa useiden asiakaskutsujen välillä ja siinä voidaan näin olleen säilyttää istunnossa tarvittavaa tietoa. (Ambler, Jewell ja Roman 2002, 82-83.)

Istuntokomponentteja voidaan passivoida, jos niitä ei käytetä tietyn ajan kuluessa, jolloin komponentin sisältämä tilatieto talletetaan pysyvään tietovarastoon, esimerkiksi kiintolevylle. Aktivoimisella tarkoitetaan istuntokomponenttien tapauksessa toimenpidettä, jolla tietovarastossa oleva tilatieto palautetaan istuntokomponentin käyttöön. Tilaton istuntokomponentti on tehokkaampi kuin tilallinen, koska tilallinen istuntokomponentti voi sisältää aktivoitavaa tilatietoa. (Adatia ym. 2001, 65.) Aktivoimisella ja passivoimisella voidaan lisätä elektronisen liiketoiminnan sovelluksen skaalautuvuutta, koska sellaisten asiakkaiden tilallisia istuntokomponetteja, joita ei ole käytetty pitkään aikaan, voidaan tallentaa tietovarastoon sen sijaan, että niitä säilytetään tietokoneen käyttömuistissa. Istuntokomponentin yleinen rakenne on kuvattuna KUVIOSSA 5.



KUVIO 5. Istuntokomponentin rakenne.

4.1.2 Istuntokomponentin lokaalit rajapinnat

EJB 1.1 –spesifikaatiossa istuntokomponentin rajapinnat jaetaan kahteen osaan: koti- ja etärajapintaan. Näitä rajapintoja käyttävä metodikutsu kulkee aina verkon yli. EJB 2.0 – spesifikaatio määrittelee kuitenkin uudet lokaalit rajapintatyypit koti- ja etärajapinnoille, jotka lisäävät metodikutsujen nopeutta.

Adatia ym. (2001, 159) mainitsevat muutamia eroavaisuuksia lokaalien ja etärajapintojen välillä. Etärajapinnat mahdollistavat *komponentin hajauttamisen* eri sovelluspalvelimille, mutta ne ovat samalla tehottomia, jos asiakkaana on toinen samassa virtuaalikoneessa sijaitseva EJB-komponentti. Lokaalit rajapinnat taas edellyttävät, että kutsuva komponentti sijaitsee samassa virtuaalikoneessa kuin kutsuttava komponentti. Lokaalit rajapinnat mahdollistavat *parametrien ja paluuarvojen välittämisen viitteenä*, jolloin olioiden koko ei vaikuta metodikutsujen nopeuteen. Edellä mainittujen seikkojen lisäksi lokaalit rajapinnat eivät vaadi *RMI-poikkeusten* (remote method invocation) käsittelemistä, koska metodikutsut eivät kulje verkon yli. (Adatia ym. 2001, 159.)

Adatia ym. (2001, 160) mainitsevat lokaalien rajapintojen käytön rajoitukseksi sen, että etärajapinnat eivät voi käyttää lokaaleja rajapintoja paluu- tai parametriarvoina. Lokaalien rajapintaluokkien avulla voidaan kasvattaa sovellusten tehokkuutta sellaisissa tapauksissa, joissa ollaan aivan varmoja siitä, että kutsuva ja kutsuttava komponentti sijaitsevat samassa virtuaalikoneessa. Ne soveltuvatkin käytettäviksi sellaisissa komponenteissa, jotka liittyvät loogisesti toisiinsa ja jotka todennäköisesti kuitenkin tulisivat sijaitsemaan samassa säiliössä.

4.1.3 Liiketoimintaprosessin toteuttaminen istuntokomponenttien avulla

Toteutettaessa liiketoimintaprosesseja EJB:n avulla pitäisi ottaa huomioon asiakkaan istunnon tyyppi, joka voi olla joko tilaton vai tilallinen, sekä suorituskykyvaatimukset.

Edellisessä kappaleessa todettiin tilattoman istuntokomponentin olevan tilallista tehokkaampi. Jos liiketoimintaprosessi koostuu vain yhdestä tehtävästä, johon kuuluu tiedon palauttaminen asiakkaalle, voi skaalautuvin vaihtoehto olla tilattoman istuntokomponentin käyttäminen.

Elektronisen liiketoiminnan sovelluksissa on usein käytössä ostoskori, johon asiakas voi lisätä tuotteita, joita hän haluaa ostaa. Tämänkaltainen toiminto vaatii tilallisen istuntokomponentin käyttämistä, koska komponentin on säilytettävä asiakkaan ostoskorin tila koko istunnon ajan. (Ambler, Jewell ja Roman 2000, 82.)

EJB mahdollistaa transaktionaalisten liiketoimintaprosessien toteuttamisen istuntokomponenttien avulla. Transaktiot voidaan joko määritellä itse tai antaa sovelluspalvelimen säiliön huolehtia transaktion hallinnasta. Säiliön kontrolloima transaktioiden hallinta on helpompi toteuttaa, mutta se on karkeajakoisempaa, koska transaktio koskee aina kokonaista metodikutsua. Tällöin pitkäkestoinen transaktionaalinen metodikutsu saattaa heikentää muiden metodien suorituskykyä. Käyttäjän itsensä määrittelemä transaktioiden hallinta puolestaan vaatii enemmän työtä, mutta sen avulla transaktioita voidaan hallita hienojakoisemmin. (Adatia ym. 2001, 379.)

Transaktioiden toteuttaminen www-selaimen lähettämien HTTP-pyyntöjen perusteella on erittäin hankalaa HTTP:n tilattoman luonteen vuoksi, koska asiakas ei välttämättä ilmoita, milloin transaktio on päättynyt (Allamaraju ym. 2001, 135). Tällaiset pitkät transaktiota ovat erittäin hankalia elektronisen liiketoiminnan sovellusten kannalta. Esimerkiksi, jos sovellus on integroituna varastojärjestelmään ja asiakas lisää tuotteen ostoskoriinsa, niin ongelmaksi muodostuu se, milloin varastosta voidaan vähentää yksi tuote. Jos varastosta ei välittömästi vähennetä tuotetta, niin asiakkaat voivat kuvitella, että kyseistä tuotetta on vielä saatavilla. Toisaalta, jos varastosta vähennetään reaaliaikaisesti tuotteita, niin organisaatio voi menettää asiakkaita, koska varasto näyttää olevan tyhjä, vaikka todellisuudessa tuotteet on vasta lisätty asiakkaiden ostoskoreihin.

Liiketoimintaprosessien toteuttaminen istuntokomponenttien avulla vaatii huolellista suunnittelua suorituskykyvaatimusten näkökulmasta. Voidaankin todeta, että mitä monimutkaisempi ja mitä useampia osapuolia prosessissa on mukana, sitä vaativampaa prosessin toteuttaminen on.

4.2 Tietosisällön mallintaminen

Liiketoimintaprosessin tarkoituksena on yleensä tiedon siirtäminen tai sen tilan muuttaminen toiseksi. Seuraavissa kappaleissa tarkastellaan tiedon tallentamista kohdekomponenttien avulla sekä siihen liittyviä EJB 2.0 spesifikaation erityispiirteitä.

4.2.1 Kohdekomponentin rakenne

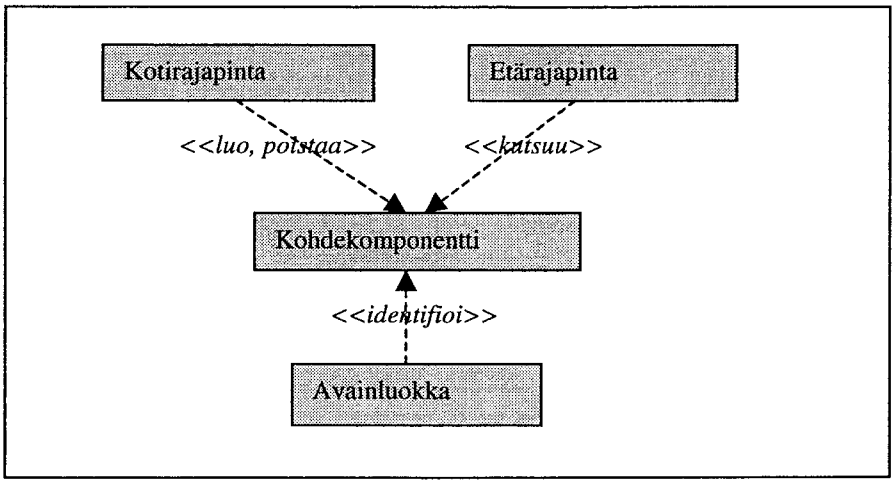
EJB-komponenttimallissa käytetään kohdekomponentteja (entity bean) tiedon tallentamiseksi pysyvään tietovarastoon. Kohdekomponentin instanssiin tallennetaan tietoa tietovarastosta siten, että komponentti kuvaa jaettua transaktionaalista tilaa. Kohdekomponentti eroaa istuntokomponentista myös siinä, että komponentin elinkaari ei määräydy asiakkaan istunnon pituuden mukaan, vaan sen määrää tietokannassa olevan tiedon olemassaolo. (Adatia ym. 2001, 24; 106.)

Kohdekomponenttien yhteys tietovarastoon voidaan toteuttaa joko säiliön tai komponentin hallitseman persistenssiyden avulla. Säiliön hallitsemassa persistenssiydessä (container managed persistence) komponentin kehittäjän ei tarvitse itse kirjoittaa tietokantakutsuja, vaan säiliö generoi ne automaattisesti. Komponentin itsensä hallitsemassa persistenssiydessä (bean managed persistence) komponentin kehittäjän täytyy itse kirjoittaa tietokantakutsut. (Adatia ym. 2001, 106.)

Kohdekomponentti identifioidaan yksilöllisen avaimen avulla (primary key), joka erottaa sen toisista kohdekomponenteista. Avain muodostaa oman luokkansa ja se

sisältää ne tietovarastossa olevien kenttien arvot, joilla kohdekomponentti identifioidaan. Säiliö käyttää sisäisesti tätä avainta, kun se hakee tietokannasta viitteet avainarvoa vastaaviin komponentteihin. (Adatia ym 2001, 107.)

Kohdekomponentilla on kaksi rajapintaa, joiden kautta sitä voidaan kutsua. Kotirajapinnan avulla kohdekomponentin kuvaamaa tietoa voidaan luoda, etsiä ja poistaa. Etärajapinnan kautta asiakas voi puolestaan kutsua kohdekomponentin metodeja. (DeMichiel, Krishnan ja Yalçinalp 2001, 109; 121.) Kohdekomponentin yleinen rakenne on kuvattu KUVIOSSA 6.



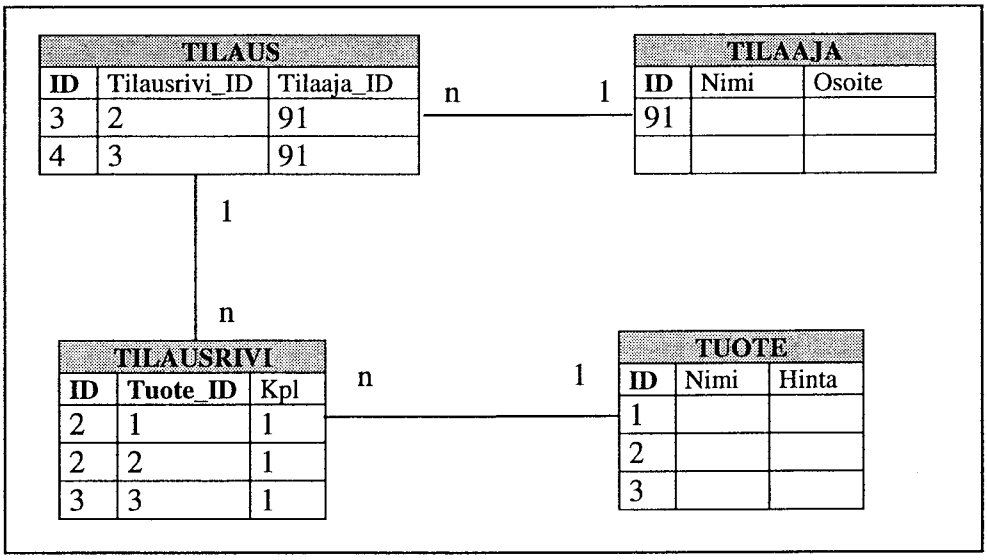
KUVIO 6. Kohdekomponentin rakenne.

4.2.2 Kohdekomponentin lokaalit rajapinnat

Kohdekomponenttien kotirajapinnat on istuntokomponenttien tapaan jaettu lokaaleihin ja etärajapintoihin, joten istuntokomponenttien kohdalla kappaleessa 3.2.3 mainitut lokaalirajapintojen ominaisuudet koskevat myös kohdekomponentteja. Lokaalit rajapinnat ovat kohdekomponenteissa hyödyllisiä etenkin niissä tapauksissa, joissa yksi istuntokomponentti hakee tietoa useammasta kohdekomponentista.

4.2.3 Kohdekomponenttien väliset suhteet

Relaatiotietokannoissa oleva tieto on yleensä normalisoitu siten, että sen redundanssi on mahdollisimman pieni. Tämä tarkoittaa sitä, että loogisesti toisiinsa liittynyt tieto on jaoteltu eri tauluihin, joita yhdistää tietty avainarvo. Esimerkiksi tilaus on liiketoiminnassa hyvin usein käytetty tieto, joka tallennetaan useisiin eri tietokannan tauluihin. Tilaus koostuu yleensä tilaajan tiedoista, tilausriveistä, jotka identifioivat tilattavat tuotteet sekä tilatuista tuotteista. KUVIOSSA 7 on esitetty tilaukseen liittyvät tietokohteet relaatiokaavion muodossa. Relaatiokomponentit sisältävät kenttiä, joista identifioiva avainarvo on lihavoitu. Kaaviota tulkitaan siten, että tilaajalla voi olla useita tilauksia, joista yhteen tilaukseen saattaa kuulua yksi tai useampia tilausrivejä. Yksi tuote voi puolestaan kuulua useampaan tilausriviin.



KUVIO 7. Tilaukseen liittyvät tietokohteet DeMichieliä, Yalçinalpia ja Krishnaa (2001, 221) mukaillen.

KUVION 7 mukaisen tiedon esittäminen EJB 1.1 –spesifikaatiossa vaatisi useita kohdekomponentteja sekä niiden hallitsemista yhtenä kokonaisuutena esimerkiksi istuntokomponentin avulla. EJB 2.0 –spesifikaatio mahdollistaa kuitenkin

kohdekomponenttien välisten monimutkaisten suhteiden määrittelyn siten, että koko rakennetta hallitaan yhdessä kohdekomponentissa, eli KUVION 7 tapauksessa tilauskomponentissa. EJB:ssä tästä käytetään nimeä säiliön hallitsemat suhteet (container managed relationships).

Adatian ym. (2001, 164-165) mukaan EJB 2.0 tukee sekä yksi- että kaksisuuntaisia kohdekomponenttien välisiä suhteita. Tämän lisäksi se tukee yhdestä-yhteen-, yhdestä-moneen- sekä monesta-moneen –suhteita. Suhteen kohteena olevaa komponenttia kuvaa lokaali rajapinta, jota hallitsee jokin toinen kohdekomponentti. (Adatia ym. 2001, 164-165.)

Säiliön hallitsemia suhteita käytetään Adatian ym. (2001, 165) mukaan kolmeen eri tarkoitukseen:

- Loogisesti yhteen liittyvien tietokohteiden käsittelemiseen
- Kohdekomponenttien elinkaaren hallintaan
- Toisiinsa liittyvien tietokohteiden hakemiseen EJB-kyselykielen avulla.

Kohdekomponenttien elinkaaria voidaan hallita suhteiden avulla siten, että kohdekomponentin hävittyä kaikki sen alaisuudessa olevat kohdekomponentit poistetaan automaattisesti. Tilausesimerkissä tämä tarkoittaisi sitä, että tilauksen poistuttua tietokannasta myös siihen liittyvät tilausrivit poistuisivat sieltä. (Adatia ym. 2001, 165.)

4.2.4 EJB-kyselykieli

EJB 1.1 –spesifikaatiossa kohdekomponenttien hakemiseen tietokannasta ei ole yhtä standardia tapaa, vaan jokaisella sovelluspalvelimen valmistajalla on siihen oma syntaksinsa. Tämä heikentää olennaisesti kohdekomponenttien siirrettävyyttä palvelimelta toiselle, koska se vaati useimmiten hakulauseen muotoilemisen uudelleen. (Adatia ym. 2001, 154.)

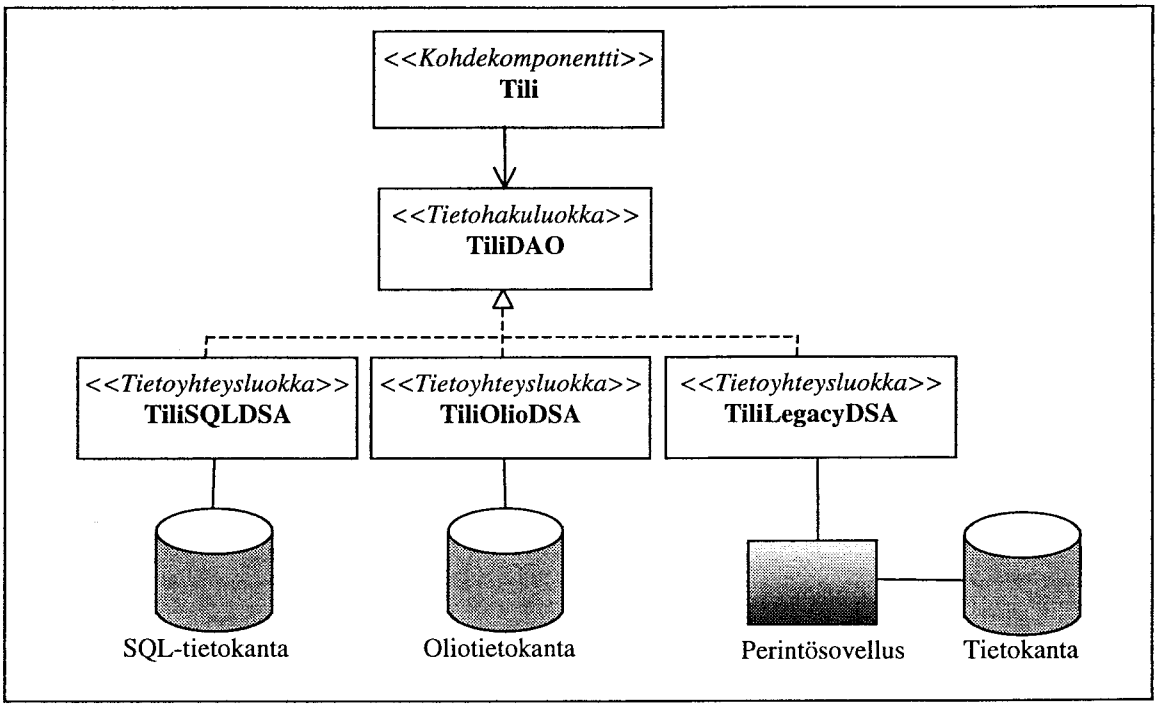
Uudemmassa EJB 2.0 –spesifikaatiossa edellisen version ongelmat on tältä osin korjattu määrittelemällä kohdekomponenttien kyselykieli, EJB QL (EJB Query Language), joka on SQL-92:n osajoukko. Kyselykieli sisältää SQL:ssä käytettyjä lauseita, kuten SELECT, FROM ja WHERE, joten SQL:ää hallitsevalle henkilölle sen opettelun ei pitäisi olla vaikeaa. Siihen kuuluu myös muutama uusi lause, jotka on kehitetty nimenomaan kohdekomponentteja ajatellen. Yksi näistä lauseista on IS EMPTY, joka ilmaisee yhdestä-moneen –suhteessa, onko suhdetta hallitsevalla komponentilla alikomponentteja. Toinen uusi lause, MEMBER OF, ilmaisee sen, kuuluuko suhteessa kohteena oleva komponentti yhdestä-moneen –suhteeseen. (Adatia ym. 2001, 169-170.)

4.2.5 Tiedon säilöminen kohdekomponentin avulla

Kohdekomponenttien tilan tallentaminen pysyvään tietovarastoon voidaan delegoida säiliön tehtäväksi tai se voidaan tehdä komponentissa. *Komponentin hallitsema persistenssiys* toteutetaan siten, että komponentin tilan lukeminen ja tallentaminen suoritetaan komponentin elinkaarta hallitsevissa metodeissa. DeMichielin, Krishnan ja Yalçinalpin (2001, 244) mukaan nämä metodit ovat ejbCreate, ejbRemove, ejbFind, ejbLoad sekä ejbStore. Metodeissa on tällöin otettava yhteys tietokantaan, luotava hakulauseet sekä huolehdittava yhteyden sulkemisesta. Metodien toteutukset voidaan kuitenkin kapseloida erillisiin luokkiin, jotka huolehtivat varsinaisen yhteyden luomisesta ja sulkemisesta sekä hakulauseen muodostamisesta.

KUVIOSSA 8 on esitetty luokkakaavio komponentin hallitseman persistenssiyden toteuttamisesta DAO- (Data Access Object) ja DSA- (Data Source Adapter) –luokkien avulla. Dahlén (2001 55-58; 64) esittää artikkelissaan DAO- ja DSA- luokkiin perustuvan arkkitehtuurin, jonka tarkoituksena on eristää tietoa säilövät luokat varsinaisesta tiedon hakemisesta, tallettamisesta sekä yhteyden muodostamisesta. Tässä mallissa DAO-luokat ovat rajapintaluokkia, jotka määrittelevät tiettyjä metodeita, joilla tietoa voidaan hakea ja käsitellä. DSA-luokat puolestaan toteuttavat nämä metodit eri tietovarastoille, jolloin yhden DAO-luokan avulla voidaan muodostaa yhteys lähes mihin tahansa tietovarastoon. Mallin tarkoituksena on eristää tietoa käsittelevä kerros

varsinaisesta tiedon lähteestä. Kohdekomponentin tapauksessa tämä tarkoittaa sitä, että vaikka tietovarasto muuttuisi SQL-pohjaisesta tietokannasta oliotietokantaan, niin itse komponenttiin ei tarvitse tehdä muutoksia. Muutokset on lokalisoitu DSA-luokkiin, joita voidaan kehittää lisää sitä mukaa kun syntyy tarve käyttää uudentyyppisiä tietovarastoja. Arkkitehtuuri mahdollistaa myös vanhojen perintösovellusten paketoimisen osaksi uutta komponenttipohjaista järjestelmää, jolloin saadaan hyödynnettyä sen tarjoamia transaktiopalveluita. (Dahlén 2001, 55-58; 64.)



KUVIO 8. Komponentin hallitseman persistenssiyden toteuttaminen DAO- ja DSA-luokkien avulla Dahlénia (2001, 57) mukaillen.

Vaikka komponentin hallitsema persistenssiys on modulaarinen ja joustava, niin sillä on kuitenkin myös huonot puolensa, jotka liittyvät ylläpidettävyyteen sekä kehittämisen nopeuteen. Sovellusten kehittäjien on hallittava erilaisia kyselykieliä sekä ylläpidettävä niitä, jos tietovarastojen skeemoissa tapahtuu muutoksia. Se on kuitenkin hyvä vaihtoehto sellaisissa tilanteissa, joissa tieto on hajallaan erityyppisissä tietovarastoissa ja tilanteissa, joissa halutaan yhtenäinen rajapinta tietokerrokseen.

Säiliön hallitsema persistenssiys on toinen tapa hallita kohdekomponentin tilatietoa. Siinä komponentin kehittäjän ei tarvitse huolehtia tietokantakutsujen luomisesta, vaan säiliö suorittaa sen automaattisesti. Komponentit ovat myös siirrettäviä ja helposti uudelleenkäytettäviä, koska tietokantayhteyttä hallitsee säiliö, joka puolestaan on erotettu komponentin toteutuksesta. Säiliön hallitsema persistenssiys soveltuukin parhaiten sellaisiin tapauksiin, joissa heterogeenisiä tietovarastoja on vähän ja joissa on tarvetta saada kehitettyä nopeasti liiketoimintalogiikkaa. (Adatia ym. 2001 106.)

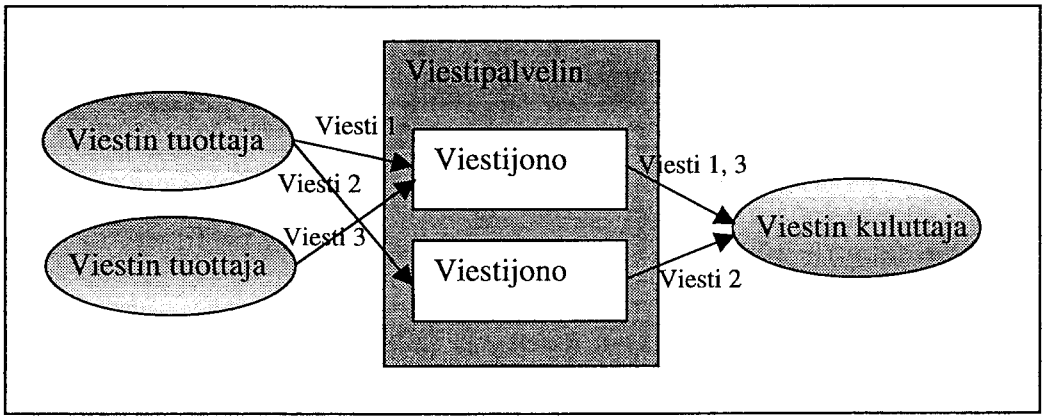
4.3 Viestin välityksen mallintaminen

Liiketoimintaprosesseissa liikkuu yleensä viestejä prosessiin kuuluvien eri tehtävien välillä. Viestit voivat sisältää tietoa esimerkiksi uudesta tilasta, johon liiketoimintakohde täytyy muuttaa tai ne voivat sisältää vastauksia pyyntöihin tai paluuarvoja. Liiketoimintaprosessin sujuvuuden kannalta viestien välitystavoissa on eroja, jotka täytyy ottaa huomioon sovellusta suunniteltaessa.

4.3.1 Asynkronisten viestien välitystapoja

Asynkronisten viestien välittämiseen on olemassa useampia tapoja ja niitä tarvitaan erilaisiin käyttötarkoituksiin. Asynkroniseen viestinvälitykseen osallistuu aina kaksi osapuolta, joista käytetään nimityksiä viestin tuottaja (message producer) ja viestin kuluttaja (message consumer). Kumpaakin osapuolta kutsutaan lisäksi viestinvälitys-asiakkaiksi, koska ne ovat asiakkaina samanarvoisia. (Adatia ym. 2001, 265-266.) Viestien tuottajien ja kuluttajien välissä toimii lisäksi palvelin, joka säilöö viestejä ja takaa sen, että viesti voidaan välittää eteenpäin, vaikka viestin kuluttaja ei olisikaan tietyllä hetkellä toimintakunnossa. Palvelin tallettaa viestejä erityisiin viestijonoihin, joista viestejä lähetetään eteenpäin tai joista niitä voidaan hakea.

Pisteestä pisteeseen (point-to-point) kommunikoinnissa jokaiselle viestille on määritelty vain yksi vastaanottaja, eli viestin kuluttaja (Ambler, Jewell ja Roman 2002, 205). KUVIOSSA 9 on kuvattu pisteestä pisteeseen viestintää, jossa yhdelle kuluttajalle lähetetään useita viestejä. Kukaan muu ei saa haltuunsa viestejä, koska ne on osoitettu vain tietylle viestin vastaanottajalle.

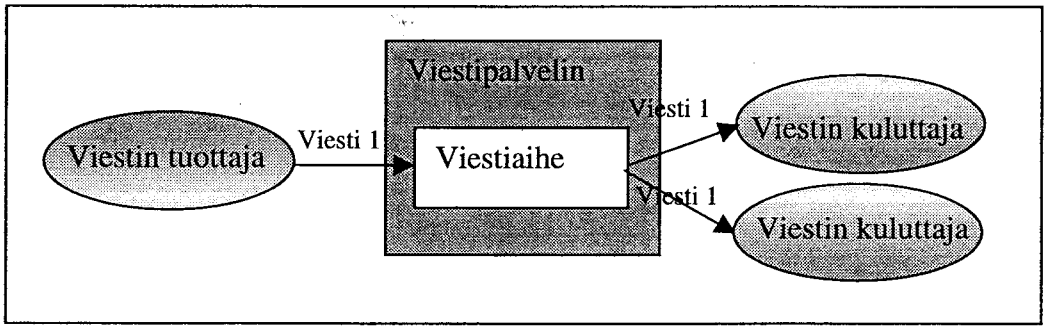


KUVIO 9. Pisteestä pisteeseen kommunikointi (Ambler, Jewell ja Roman 2002, 205).

Pisteestä pisteeseen viestintä koostuu kolmesta vaiheesta (Adatia ym. 2001, 267):

- Viestin kuluttaja rekisteröi itsensä tiettyyn viestijonoon
- Viestin tuottaja lähettää viestin kyseiseen jonoon viestipalvelimella
- Viestipalvelin välittää viestin viestijonosta siihen rekisteröityneelle kuluttajalle, kun kuluttajalla on mahdollisuus käsitellä viestiä.

Julkaisu ja tilaus –viestinvälitys on toinen asynkronisen kommunikoinnin muoto (KUVIO 10), jossa viestin tuottaja julkaisee tiettyyn viestiaiheeseen (topic) kuuluvan viestin. Viestin kuluttajat voivat ilmoittautua tietyn viestiaiheen lukijoiksi ja yhteen viestiaiheeseen voi tällöin kuulua useita kuluttajia. Kyseessä on typillinen yhdestä-moneen –suhde eli niin kutsuttu broadcast-malli.



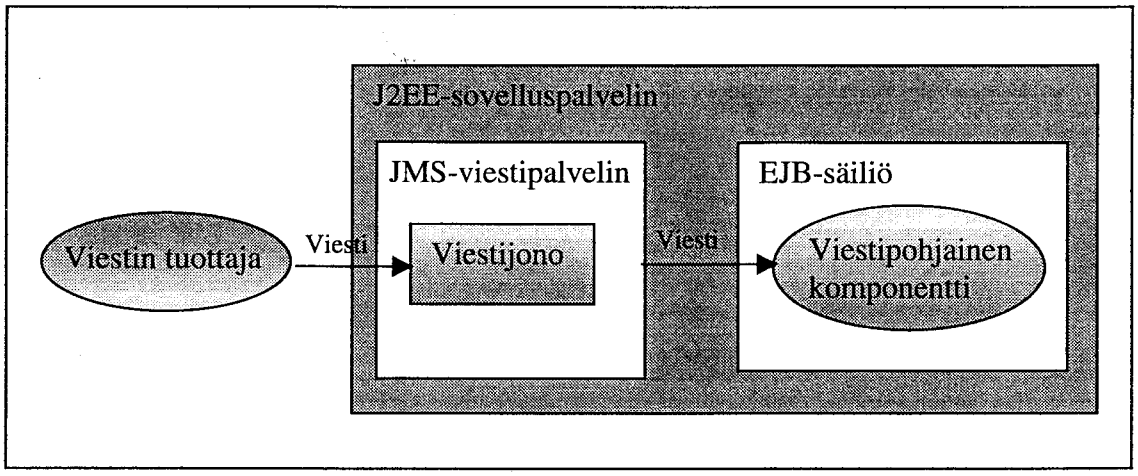
KUVIO 10. Julkaisu ja tilaus –viestinvälitys (Ambler, Jewell ja Roman 2002, 205).

Julkaisu ja tilaus –viestinvälitys tapahtuu myös kolmessa vaiheessa (Adatia ym. 2001, 268):

- Viestin kuluttajat rekisteröivät itsensä tietyn viestiaiheen kuuntelijoiksi
- Viestin tuottaja lähettää viestin tiettyyn viestiaiheeseen palvelimella
- Viestipalvelin toimittaa tietyn aiheen viestin kaikille siihen rekisteröityneille kuluttajille.

4.3.2 Viestipohjaisen komponentin rakenne

EJB 2.0 –komponenttimallin kolmannen komponenttityypin muodostaa viestipohjainen komponentti (message-driven bean). Sen tehtävänä on toimia asynkronisten viestien kuluttajana. Asiakkaan lähettäessä viestin JMS-jonoon tai –aiheeseen EJB-säiliö herättää kyseistä jonoa tai aihetta kuuntelevan viestipohjaisen komponentin, joka tämän jälkeen suorittaa tietyn tehtävän. (DeMichiel, Krishnan ja Yalçinalp 2001, 311.) KUVIOSSA 11 kuvataan viestin lähettäminen viestipohjaiselle komponentille.



KUVIO 11. Viestin välittäminen viestipohjaiselle komponentille (Adatia ym. 2001, 275).

DeMichiel, Krishnan ja Yalçinalp (2001, 311) toteavat, että viestipohjaisilla komponentilla ei ole koti- eikä etärajapintoja, joten asiakkaana toimiva viestin tuottaja ei ole missään vaiheessa suoraan yhteydessä siihen. Komponentti ei myöskään säilytä minkäänlaista tilatietoa, joten eri asiakkaiden lähettämiä viestejä voi käsitellä sama viestipohjaisen komponentin instanssi. (DeMichiel, Krishnan ja Yalçinalp 2001, 311.)

Adatian ym. (2001, 275) mukaan viestin lähettäminen koostuu neljästä vaiheesta:

- Asiakas etsii viestijonon tai –aiheen referenssin JNDI-palvelimelta
- Asiakas lähettää viestin viestinvälityspalvelimelle
- Viestinvälityspalvelin kutsuu viestipohjaisen komponentin onMessage-metodia
- EJB-säiliö huolehtii viestipohjaisen komponentin instanssin luomisesta ja onMessage-metodin kutsumisesta.

Kuten edellisistä vaiheista käy ilmi, viestipohjaisen komponentin rakenne on hyvin yksinkertainen, koska se määrittelee vain yhden metodin, onMessage, jota kutsutaan viestin saapuessa. Metodi ottaa vastaan yhden parametrin, joka sisältää viestin. (DeMichiel, Krishnan ja Yalçinalp 2001, 315.)

JMS-viestipalvelimelle lähetettävät viestit voivat olla viittä eri tyyppiä: tekstipohjaisia, avain-arvopariin perustuvia, binäärisiä, primitiivityyppejä sisältäviä sekä olioita (Adatia ym. 2001, 271). Viestityyppien tarkempi erittely on esitetty TAULUKOSSA 2.

TAULUKKO 2. JMS-viestien tyypit Adatiaa ym. (2001, 271) mukaillen.

Tiedon tyyppi	JMS-luokka	Käyttötarkoitus
Teksti	TextMessage	Tekstipohjaiset viestit, esimerkiksi XML
Avain-arvopari	MapMessage	Tietyn avaimen perusteella yksilöitävät tiedot
Binäärivirta	BytesMessage	Binääritiedostojen siirtäminen
Primitiivityypit	StreamMessage	Javan primitiivityyppien lähettäminen
Olio	ObjectMessage	Kompleksit datatyyppit, serialisoituvat Java-oliot

4.3.3 Tiedon välittäminen viestipohjaisen komponentin avulla

Asynkroninen kommunikointi on joissakin tapauksissa huomattavasti tehokkaampaa ja liiketoiminnallisesti järkevämpää kuin välittömästi vastauksen antava synkroninen kommunikointi. Allamaraju ym. (2001, 141) määrittelevät teknisiä tarpeita asynkronisten viestien käytölle elektronisen liiketoiminnan sovellusten näkökulmasta. He jakavat tarpeet kolmeen luokkaan: saatavuuteen, vasteaikaan ja resurssirajoituksiin.

Tietyissä tilanteissa, joissa pääjärjestelmä on riippuvainen osajärjestelmistä, voi asynkroninen kommunikointi huomattavasti parantaa palvelun *saatavuutta*. Tällainen tilanne saattaa tulla eteen esimerkiksi asiakkaan lähettäessä elektronista tilausta. Jos jokin osajärjestelmä sattuisi pettämään, niin synkronisen kommunikoinnin tapauksessa pääjärjestelmä ei voisi ottaa vastaan tilausta. Asynkroninen järjestelmä sen sijaan mahdollistaisi asiakkaan tilauksen tallettamisen järjestelmään. (Allamaraju ym. 2001, 137-138.) Palvelun saatavuudella on tässä tapauksessa kriittinen merkitys liiketoiminnan kannalta, koska asiakkaan tilauksen lähettämisen epäonnistuminen aiheuttaisi todennäköisesti asiakkaan menettämisen jollekin toiselle yritykselle.

Vasteaika voi muodostua merkittäväksi tekijäksi liiketoimintaprosessissa, etenkin jos se muodostuu liian pitkäksi. Tilauksen yhteydessä synkronisesti tehtävä luottotietojen tarkistus voi olla aikaa vievä operaatio, mikä saattaa heikentää asiakkaan kokemaa vasteaikaa. (Allamaraju ym. 2001, 139.) Etenkin elektronisessa liiketoiminnassa asiakas voi hyvin nopeasti turhautua, jos hän luulee järjestelmän kaatuneen pitkän vasteajan vuoksi. Tällöin hän saattaa keskeyttää transaktion ja etsiä vaihtoehtoisia palveluiden tarjoajia.

Myös *teknisten resurssien rajoitteet* voivat heikentää synkronisten järjestelmien suorituskykyä, jos järjestelmällä on ennalta arvaamaton määrä käyttäjiä, eikä se skaalaudu kuormituksen mukaan (Allamaraju ym. 2001, 140-141).

Kaikissa edellä mainituissa tapauksissa asynkronisella kommunikoinnilla voitaisiin parantaa asiakkaan kokemaa palvelun laatua. Asynkroninen kommunikointi ei kuitenkaan ole täysin ongelmatonta, koska viestipalvelin voi kaatua tai transaktio keskeytyä, jolloin asiakkaalle ei välttämättä voida enää välittää viestiä tapahtuneesta.

Viestipalvelimen kaatumista varten viesteille voidaan antaa persistenssiysmäärityksiä, jotka aiheuttavat viestien tallentumisen pysyvään tietovarastoon palvelimella. Palvelimen noustessa ylös, viesti luetaan tietovarastosta ja säilytetään muistissa tietyn ajan, jonka jälkeen viesti tuhotaan. (Adatia ym. 2001, 272.) Viestien tallentaminen pysyvään tietovarastoon vaikuttaa kuitenkin haitallisesti järjestelmän suorituskykyyn, joten viestin pysyvyyden määrittelyssä tulee huolellisesti punnita suorituskyky-vaatimuksia sekä viestin perille saamisen tärkeyttä.

4.4 Yhteenveto

Luvussa tarkasteltiin komponenttiperusteista sovelluskehitystä EJB-komponenttimalliin perustuen ja tutkittiin sitä kuinka eri komponenttityypit tukevat liiketoimintaprosessien mallintamista ja kehittämistä sekä niitä ongelmia, joita komponenttiperusteisella tekniikalla voidaan ratkaista. EJB 2.0 –komponenttimalli tukee kolmen tyyppisiä

komponentteja, jotka voivat olla istunto-, kohde- tai viestipohjaisia. Näiden komponenttityyppien avulla voidaan toteuttaa yksinkertaisia liiketoimintaprosesseja.

Istuntopohjaisilla komponenteilla voidaan mallintaa liiketoimintaprosessia kokonaisuutena sekä hyödyntää EJB-komponenttimallin tarjoamia transaktiopalveluita. Tilalliset istuntopohjaiset komponentit mahdollistavat metodikutsujen välisen tilan tallentamisen, kun taas tilattomat komponentit eivät säilytä asiakkaan tilatietoa lainkaan. Komponenttityyppien nopeuteen ja toiminnallisuuteen liittyvät erot vaikuttavat niiden käyttötapoihin.

Kohdepohjaiset komponentit mallintavat liiketoimintaprosesseissa käsiteltäviä liiketoimintakohteita eli tietoa. Niiden avulla tietoa saadaan säilöttyä pysyvään tietovarastoon siten, että se on käytettävissä vaikka sovelluspalvelin välillä sammutettaisiin. EJB 2.0 –spesifikaatio tarjoaa lisäksi uuden kohdekomponenttien kyselykielen EJB QL:n, jonka avulla komponenteista tulee entistä siirrettävämpiä. Sekä istunto- että kohdepohjaiset komponentit mahdollistavat saman virtuaalikoneen sisällä tehtävät metodikutsut niin sanottujen lokaalien rajapintojen kautta. Tällöin metodien parametrit voidaan välittää referensseinä, jolloin suorituskyky kasvaa.

Viestipohjaisilla komponenteilla voidaan mallintaa asynkronisia viestejä sisältäviä prosesseja. Asynkronisuus lisää liiketoimintaprosessien sujuvuutta etenkin asiakkaan näkökulmasta, koska hänen ei tarvitse jäädä odottamaan prosessin tuloksena syntyvää paluuarvoa. Tämä aiheuttaa myös ongelmia, koska asiakas ei välttämättä heti saa tietää prosessissa tapahtuneesta poikkeuksesta, jolloin transaktioiden hyödyntäminen vaikeutuu. Viestipohjaisten komponenttien toiminta perustuu viestipalvelimella oleviin jonoihin sekä niihin lähetettäviin viesteihin. Komponentit muistuttavat tilattomia istuntopohjaisia komponentteja, mutta niillä ei ole koti- eikä etärajapintoja, joita asiakas voisi kutsua. Istuntopohjaiset komponentit ovatkin täysin EJB-säiliön kontrollissa.

5 VERKKOPALVELUT KOMPONENTTEINA

Liiketoimintaprosessien automatisoinnissa on pitkälti kyse erillisten järjestelmien integroinnista keskenään. Tämän kaltainen tilanne tulee eteen etenkin elektronisen liiketoiminnan sovelluksissa, joissa yritysten operationaaliset järjestelmät pitäisi saada keskustelemaan modernien verkkopohjaisten sovellusten kanssa. Tällöin puhutaan yleensä sisäisestä integroinnista, koska sovellukset toimivat yhden organisaation sisällä. Monet liiketoimintaprosessit ovat kuitenkin hajautettuja, jolloin prosessien toteuttamiseen saattaa osallistua ulkopuolisia tahoja ja järjestelmiä. Tällaisten reaalimaailmassa kitkattomasti toimivien prosessien automatisoiminen ja siirtäminen tietoverkkoihin tulee olemaan suuri haaste tulevina vuosina, koska uusia prosesseja pitäisi pystyä luomaan yhä dynaamisemmin. Samalla pitäisi kuitenkin kyetä hallitsemaan hajautettujen järjestelmien mukanaan tuoma monimutkaisuuden lisääntyminen.

Tässä luvussa tarkastellaan tarkemmin verkkopalveluteknologiaa (web-service), jonka tarkoituksena on ratkaista edellä mainittuja ongelmia. Aluksi määritellään verkkopalvelun käsite ja esitetään verkkopalveluarkkitehtuurin toiminta. Tämän jälkeen tutkitaan tarkemmin verkkopalveluarkkitehtuurin toteuttamisen mahdollistamia teknologioita. Lopuksi tarkastellaan niitä ongelmia, joita liiketoimintaprosessien toteuttaminen verkkopalveluilla aiheuttaa.

5.1 Verkkopalvelu

Edellisessä luvussa tarkasteltiin komponentteja sekä esitettiin niistä seuraavanlainen määritelmä: komponentti on rajapinnoilla erotettu ohjelman osa, joka voidaan vaihtaa toiseen saman rajapinnan toteuttavaan komponenttiin ilman, että asiakaspäähän tarvitsee tehdä muutoksia. Komponentin käsitteeseen liittyy siten mahdollisuus korvata se toisella samankaltaisella komponentilla, joka suorittaa halutun toiminnallisuuden tehokkaammin, turvallisemmin tai edullisemmin. Komponentti voidaan siis nähdä

tavallista oliota abstraktimpana käsitteenä, koska olio itsessään tarjoaa tietyt rajapinnat, joita voidaan kutsua. Suurin ero komponentteihin nähden on siinä, että vaihdettaessa olio toiseen, on myös sen rajapinta vaihdettava, koska se sijaitsee oliossa itsessään. Seuraavaksi tarkastellaan perinteisiä komponenttimalleja abstraktimpaa verkkopalvelua.

5.1.1 Verkkopalvelun määritelmä

Szyperski (2001) pohtii kirjoittamassaan artikkelissa komponentin ja verkossa tarjottavan sovelluksen (verkkopalvelu) eroja ja päätyy siihen lopputulokseen, että niillä on hyvin paljon yhteistä. Siinä missä komponentti on olion abstraktio, niin verkkopalvelu voidaan nähdä komponenteilla rakennetun sovelluksen abstraktiona; kummallakin on tarkasti määritelty rajapinta ja dynaamisesti vaihdettava toteutusosa. Samanlaiseen analogiaan verkkopalveluiden ja komponenttiteknologian välillä ovat myös päätyneet Matsuda, Yokoyama ja Yoshida (2002, 215), jotka vertaavat verkkopalvelurajapintaa CORBA:ssa käytettävään ohjelmointikieliriippumattomaan IDL:ään (Interface Definition Language).

Elektronisen liiketoiminnan näkökulmasta verkkopalvelu tarjoaa uusia mahdollisuuksia dynaamisten organisaatioiden välisten sovellusten toteuttamiseen. Organisaatioiden sovellukset ovat perinteisesti olleet yksityisiä, koska niiden toimintaa ei ole haluttu paljastaa ulkopuolisille, puhumattakaan sellaisten rajapintojen paljastamisesta, joilla sovellusten tarjoamia palveluita voitaisiin hyödyntää. Tämä puute on aiheuttanut sen, että kolmansien osapuolten sovelluksia dynaamisesti hyödyntäviä palveluita ei ole voitu kehittää. Yhtenä syynä palveluiden vähyyteen on ollut myös yhtenäisen sovellusrajapinnan puuttuminen sekä erilaiset laitteisto- ja ohjelmistostandardit, joiden integroiminen vaatisi räätälöintiä, mikä taas tekisi palveluista kaikkea muuta kuin dynaamisia.

Kirjallisuudessa ei ole esitetty selkeää määrittelyä verkkopalvelun käsitteelle, koska aiheen tuoreuden vuoksi sitä ei vielä ole tutkittu kovin paljon. Casati ja Shan (2001, 143-144) puhuvat kuitenkin e-palvelun (e-service) käsitteestä ja tarkoittavat sillä

dynaamisesti verkon yli tarjottavia palveluita, joita voidaan koostaa toisista e-palveluista. Dynaamisuus tarkoittaa heidän mukaansa sitä, että palvelut voivat automaattisesti sopeutua liiketoimintaprosesseissa tapahtuviin muutoksiin, ilman että käyttäjä huomaa sitä. E-palveluiden pitäisi myös voida sopeutua käyttäjän tarpeiden sekä verkossa tarjottavien muiden palveluiden mukaan. (Casati ja Shan 2001, 143-144.)

Adatia ym. (2001, 999) määrittelevät verkkopalvelun erilliseksi toiminnalliseksi osaksi, joka voidaan paikantaa ja jota voidaan kutsua tietoverkkojen yli standardilla tavalla. Verkkopalveluita voidaan lisäksi yhdistää muiden verkkopalveluiden kanssa innovatiivisten prosessien ja arvoketjujen luomiseksi. (Adatia ym. 2001, 999.)

Heuvel, Papazoglou ja Yang (2002, 125) puolestaan toteavat verkkopalvelun olevan itsensä määrittelevä sovellus, joka toimii tietoverkossa. Verkkopalvelut voivat heidän mukaansa osallistua kokonaisten liiketoimintatransaktioiden toteuttamiseen, jolloin verkkopalveluita on kyettävä koostamaan joustavasti. (Heuvel, Papazoglou ja Yang 2002, 125.)

Verkkopalvelukäsitteen lanseerannut Hewlett-Packard määrittelee verkkopalvelun sellaiseksi verkossa tarjottavaksi palveluksi tai resurssiksi, jota voivat kutsua ihmiset, organisaatiot ja laitteet. Useita verkkopalveluita voidaan myös liittää yhteen automaattisesti lähes minkä tahansa tehtävän tai transaktion suorittamiseksi. (Amor 2000, 568-569.)

Boubez ym. (2001, 9) puolestaan määrittelevät verkkopalvelun alusta- ja toteutusriippumattomaksi sovelluskomponentiksi, joka voidaan kuvata erityisellä kuvauskielellä ja julkaista palvelurekisteriin. Verkkopalveluita voidaan hakea rekisteristä standardin hakumekanismin avulla, minkä jälkeen sitä voidaan kutsua rajapinnan kautta. Verkkopalveluita voidaan heidän mukaansa myös koostaa muista verkkopalveluista. (Boubez ym. 2001, 9.)

Edellä esitettyjen määritelmien pohjalta verkkopalvelusta voidaan löytää seuraavanlaisia ominaisuuksia:

- Se on itsenäinen tietoverkoissa toimiva sovellus
- Sitä voidaan etsiä palvelurekistereistä
- Sitä voidaan kutsua standardirajapintojen kautta
- Siitä voidaan dynaamisesti koostaa uusia palveluita
- Se sopeutuu käyttäjän tarpeisiin.

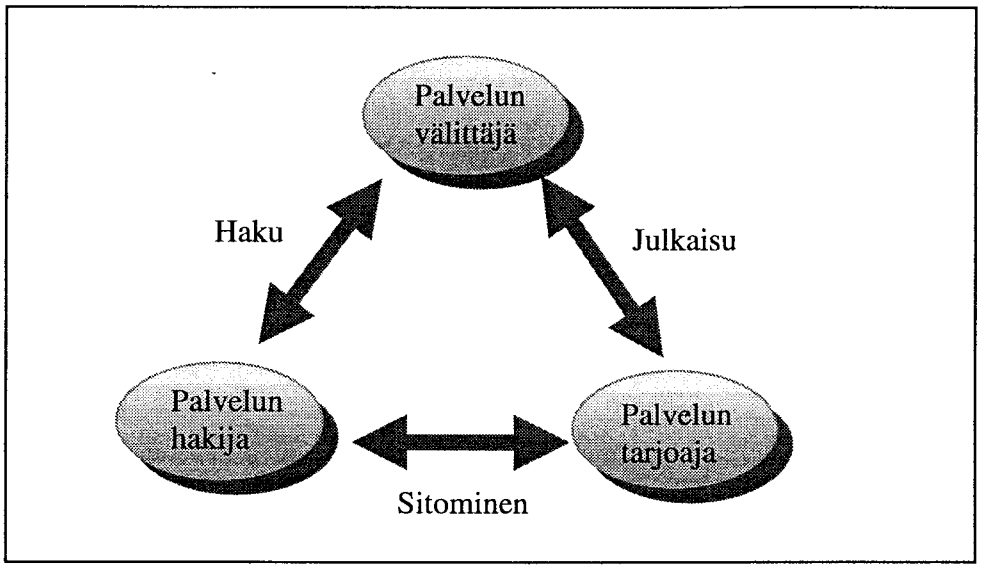
5.1.2 Verkkopalveluarkkitehtuuri

Verkkopalvelu on itsenäinen sovellus, mutta se tarvitsee ympärilleen infrastruktuurin, joka mahdollistaa palveluiden hakemisen ja kutsumisen. Casati ja Shan (2001, 144-145) kutsuvat tällaista infrastruktuurin ESP:ksi (e-service platform) ja sen tehtävänä on turvata verkkopalveluiden kehittämisen, asentamisen ja turvallisen palvelun tarjoamisen asiakkaille. He mainitsevat infrastruktuureista esimerkkeinä Hewlett-Packardin e-speak:n, Sun Microsystemsin JINI:n sekä Microsoftin NET:n. Kaikilla näillä alustoilla on yhteisinä ominaisuuksina palveluiden rekisteröiminen, palveluista tiedottaminen sekä niiden hakeminen. (Casati ja Shan 2001, 144-145.)

Adatian ym. (2001, 1000) mukaan verkkopalveluiden infrastruktuuri voidaan kuvata arkkitehtuurimallilla, jossa verkkopalveluiden käyttäjät on jaettu kolmeen rooliin: palvelun hakijaan, palvelun tarjoajaan ja palvelun välittäjään. *Palvelun hakija* on osapuoli, jonka tavoitteena on löytää ja käyttää verkkopalvelua. *Palvelun tarjoaja* julkaisee verkkopalvelun kuvauksen yhteen tai useampaan rekisteriin, josta palvelun hakija voi sen löytää. *Palvelun välittäjän* tehtävänä on puolestaan saattaa yhteen palvelun hakija ja palvelun tarjoaja. (Adatia ym. 2001, 1000.)

KUVIOSSA 12 on kuvattuna verkkopalveluarkkitehtuurin eri osapuolet ja niiden tehtävät. Palvelun tarjoaja voi *julkaista* verkkopalvelunsa tiedot välittäjän tarjoaman rajapinnan kautta. Julkaisu sisältää tiedot verkkopalvelun tarjoamasta palvelusta sekä siitä, miten verkkopalvelua voi kutsua. Palvelun hakija voi *hakea* välittäjältä tietoa julkaistuista verkkopalveluista erilaisten hakukriteerien perusteella. Hakijan löydettyä

sopivan verkkopalvelun, hän voi kutsua verkkopalvelua *sitomalla* sen omaan sovellukseensa välittäjältä löydetyn osoitetiedon perusteella. (Adatia ym. 2001, 1001.)



KUVIO 12. Verkkopalveluarkkitehtuuri Adatian ym. (2001, 1000) mukaan.

5.2 Verkkopalveluiden teknologioita

Verkkopalvelun perimmäisenä ideana on hajautettujen järjestelmien mukanaan tuoma ongelma; kuinka löytää järjestelmän eri osat ja niiden tarjoamat palvelut (Boubez ym. 2001, 11). Tästä näkökulmasta katsottuna verkkopalvelut eivät ole mikään mullistava keksintö. Esimerkiksi verkkopalveluiden ja EJB-komponenttien hakemisen välillä voidaan löytää analogia: JNDI-rekisteri toimii resurssipalvelimena, josta RMI:n avulla voidaan kutsua palveluita eli EJB-komponentteja (Adatia ym. 2001, 1013). Samanlainen analogia voidaan löytää monien muiden hajautettujen järjestelmien, kuten DCOM:n ja CORBA:n, osalta. Se mikä tekee verkkopalvelun erikoiseksi teknisestä näkökulmasta on se, että palveluiden haku ja käyttäminen toteutetaan noudattamalla yleisesti hyväksyttyjä standardeja (Boubez ym. 2001, 111).

Emmerich, Finkelstein ja Piccinelli (2001, 170) esittävät ehtoja, joiden täyttyminen mahdollistaa organisaation palveluiden tarjoamisen verkkopalveluina. Ensinnäkin, verkkoinfrastruktuurin pitäisi tarjota kommunikointiväylät, jotka mahdollistavat automaattisen tietojen vaihdon. Toiseksi, protokollat palveluiden hakemiseksi pitäisi määritellä ulkopuolelta, jotta eri osapuolet voisivat automaattisesti etsiä palveluita. Näiden ehtojen täytyttyä verkkopalveluita pitäisi vielä voida koostaa siten, että verkkopalvelun tarjoamiseen osallistuu useita osapuolia. (Emmerich, Finkelstein ja Piccinelli 2001, 170.)

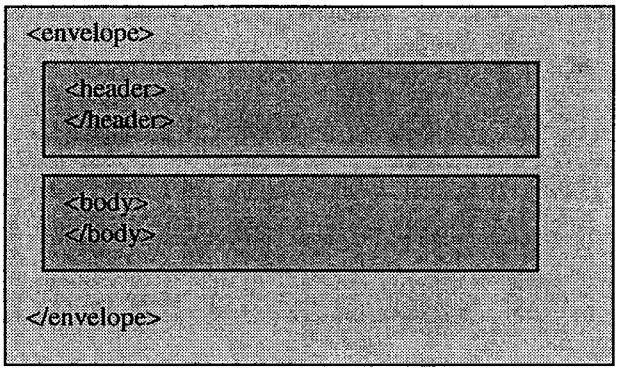
5.2.1 Palvelun kutsuminen

Verkkopalvelun ehkä tärkein osa-alue on palvelun kutsuminen, koska se tapahtuu joka kerta, kun verkkopalveluarkkitehtuuriin kuuluva osapuoli ottaa yhteyttä toiseen osapuoleen. Verkkopalveluiden laajamittainen käyttöönotto edellyttää standardeja kutsumekanismeja samalla tavalla kuin Internetin räjähdysmäinen kasvu edellytti HTTP-protokollan yleismaailmallista käyttämistä HTML-sivujen lataamiseksi.

SOAP (Simple Object Access Protocol) syntyi vuonna 1998, kun Microsoftin ryhtyi kehittämään tekniikkaa, joka mahdollistaisi etäkutsujen tekemisen HTTP:tä käyttävissä tietoverkoissa. SOAP on sen jälkeen standardoitu W3C:ssä (World Wide Web Consortium), mikä luo sille edellytykset laajalle käyttöönotolle. (Boubez ym. 2001, 118.)

SOAP perustuu hajautetussa arkkitehtuurissa tapahtuvaan kevyeen pyyntöjen välittämiseen asiakkaan ja palvelimen välillä. Vaikka pyyntöjä välittävänä protokollana voi periaatteessa olla mikä tahansa protokolla, niin käytännössä välittämiseen käytetään HTTP:tä. Pyynnöt välitetään XML-dokumenteissa, jotka koostuvat kolmesta osasta: kuoresta, otsikosta ja rungosta. *SOAP-kuoressa* voidaan määritellä XML-dokumentissa käytettäviä nimiavaruustietoja sekä erityisiä attribuutteja, joilla määritellään muun muassa käytettävä merkistö. Kuori pitää sisällään valinnaisen otsikko-osan sekä pakollisen runko-osan. *Otsikko-osassa* voidaan määritellä esimerkiksi autentikointi- tai

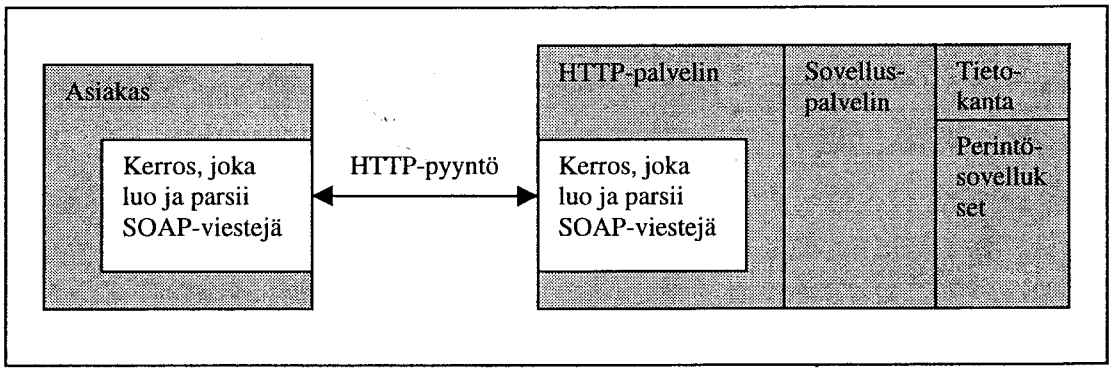
transaktiotietoja. *SOAP:n runkoa* käytetään tiedon välittämiseen asiakkaan ja palvelimen välillä ja se voi sisältää viestin tai mahdollisen virheilmoituksen. (Box ym. 2000.) KUVIOSSA 13 on esitetty SOAP-viestin rakenne sekä elementtien oikeat nimet.



KUVIO 13. SOAP-viestin XML-rakenne Adatian ym. (2001, 1002) mukaan.

SOAP:n käyttökelpoisuutta hajautettujen järjestelmien yhdistämisessä puoltaa kaksi seikkaa. Ensinnäkin, useat etäkutsuja käyttävät teknologiat, kuten DCOM, vaativat paljon kaistaleveyttä viestin välitykseen, mikä laskee viestin hyötysuhdetta. Toiseksi, etäkutsut soveltuvat huonosti sellaisiin sovelluksiin, jotka ovat palomuurien takana, koska ne vaativat porttien aukaisemista organisaation järjestelmiin, mikä ei yleensä ole suotavaa. Lähes kaikki palomuurit laskevat kuitenkin HTTP-pyynnöt palomuurin läpi, mikä tekee SOAP:n käyttämisen yhdessä HTTP:n kanssa houkuttelevaksi. (Damiani ym. 2001, 504-505.)

SOAP:n avulla voidaan periaatteessa mistä tahansa HTTP:tä ymmärtävästä sovelluksesta tehdä verkkopalvelu. Tällä saavutetaan se hyöty, että olemassa olevia sovelluksia voidaan hyödyntää verkkopalveluarkkitehtuurissa. KUVIOSSA 14 on kuvattu olemassa olevan sovelluksen muuttaminen verkkopalveluksi. Sovelluksen koolla ei ole merkitystä, joten pienimmillään se voi olla porttia 80 kuunteleva yksinkertainen sovellus ja suurimmillaan monikerroksinen toiminnanohjausjärjestelmä.



KUVIO 14. Olemassa olevan sovelluksen muuttaminen SOAP:ia käyttäväksi verkkopalveluksi.

5.2.2 Palvelun kuvaus

SOAP-pyyntöä tehtäessä on tiedettävä verkkopalvelua tarjoavan koneen IP-osoite, palvelupolku sekä portti, jotta oikea palvelu voidaan herättää. Verkkopalveluiden yksi oleellinen ominaisuus on kuitenkin dynaamisuus, mikä tarkoittaa sitä, että verkkopalvelun tarjoaja voidaan automaattisesti generoida SOAP-pyyntöön. Palvelun rajapintakuvaus tarjoaa tietoja muun muassa verkkopalvelun syötteistä, tulosteista sekä siitä osoitteesta, josta verkkopalvelua voidaan kutsua. Rajapintakuvausta voidaan eräällä tavalla verrata uudelleenkäytettävän komponentin rajapintaan, joka määrittelee ne syötteet ja tulosteet, joita komponentti voi käsitellä, sekä niiden merkityksen. Palvelun rajapintakuvausta käytetään myös verkkopalveluiden etsimiseen palvelurekistereistä.

WSDL (Web Service Description Language) on World Wide Web Consortiumin määrittelemä standardi verkkopalveluiden kuvaamiselle XML:llä. WSDL kuvaa tietoverkkojen päätepisteitä sekä niissä tarjottavia palveluita siten, että asiakkaat voivat automaattisesti generoida kutsuja kyseisiin palveluihin. WSDL määrittelee verkkopalvelun seitsemällä eri XML-elementillä, jotka ovat: porttityyppi, operaatio, viesti, tyypit, sitominen, portti ja palvelu. (Christensen ym. 2001.)

Porttityyppi (portType) kuvaa abstraktilla tasolla ne operaatiot, joita päätepiste tarjoaa, eli se on verkkopalvelun rajapintakuvaus. Yleensä porttityyppejä on vain yksi kutakin

verkkopalvelua kohti, mutta yksi porttityyppi voi sisältää useita kuvauksia niistä operaatioista, joita se tarjoaa. (Boubez ym. 2001, 329; 333.)

Operaatio (operation) kuvaa yhtä verkkopalvelun tarjoamaa metodia ja se sisältää parametrien ja paluuarvon tyypit. Operaatioita on olemassa neljää eri tyyliä sen mukaan kuinka palvelua kutsutaan. Pyyntö-vastaus –tyylissä (request-response operation) asiakas tekee pyynnön, johon hän voi odottaa saavansa vastauksen. Yksisuuntaisessa pyynnössä (one-way operation) asiakas pyytää vain verkkopalvelun tilan muuttamista, eikä odota mitään vastausta palvelimelta. HTTP-protokollassa yksisuuntaisia pyyntöjä ei voida kuitenkaan toteuttaa. Notifikaatio-pyynnössä (notification operation) palvelimelta lähtee viesti asiakkaalle tietyn ennalta määrätyn ehdon täytyttyä. Tällainen ehto voisi olla esimerkiksi osakekurssin ylitettyä tietyn tason. Push-vastaus (solicit-response) on muuten samanlainen kuin notifikaatio-pyyntö, mutta palvelin jää odottamaan lähettämäänsä viestiin vastausta. (Boubez ym. 2001, 333-337.)

Viesti (message) määrittelee ne tiedot, joita voidaan välittää palvelua kutsuvan ja sitä tarjoavan osapuolen välillä. Näihin viesteihin kuuluvat muun muassa virheilmoitukset sekä syötteet ja tulosarvot. (Boubez ym. 2001, 338.)

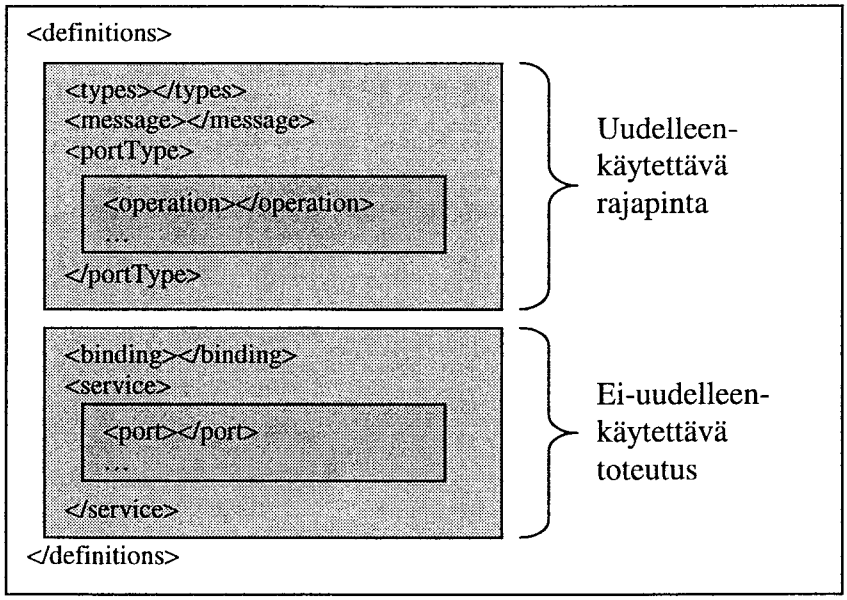
Tyypit (types) kuvaavat verkkopalvelun ymmärtämät tietotyypit. Oletustyyppi-järjestelmänä WSDL käyttää XML Schemaa (XSD). Tyyppijärjestelmä mahdollistaa myös esimerkiksi Java-tyyppien käytön verkkopalveluissa, mutta tällöin menetetään verkkopalveluille ominainen ohjelmointikieliriippumattomuus. (Boubez ym. 2001, 339-340.)

Sitominen (binding) kuvaa konkreettisesti tietyssä operaatiossa käytettävää protokollaa. Yhtä porttityyppiä kohden voi olla useita sitomiselementtejä, jotka kuvaavat niitä protokollia, joita käyttäen verkkopalvelua voidaan kutsua. Yksinkertaisuuden vuoksi yhtä porttityyppiä kohti kannattaa käyttää yhtä sitomiselementtiä. (Boubez ym. 2001, 342-343.)

Portti (port) kuvaa yhden verkkopalvelun päätepisteen ja se koostuu sitomistiedosta sekä konkreettisesta verkko-osoitteesta. Päätepiste kuvaa käytännössä sitä palvelinta, jossa verkkopalvelu sijaitsee ja johon palvelupyyntö pitää lähettää. (Boubez ym. 2001, 350.)

Palvelu (service) kuvaa yhteen kuuluvien päätepisteiden joukkoa. Palveluelementtiä voidaan käyttää esimerkiksi kokoamaan yhteen sellaisia porttielementtejä, jotka viittaavat samaan verkkopalveluun, mutta jota voidaan kutsua erilaisten protokollien, kuten SOAP/HTTP tai SOAP/SMTP avulla. (Boubez ym. 2001, 342-343.)

KUVIOSSA 15 on kuvattu WSDL-dokumentin XML-rakenne. Se voidaan jakaa uudelleenkäytettävään ja ei-uudelleenkäytettävään osaan. Uudelleenkäytettävä osa koostuu sellaisista tiedoista, joita voidaan käyttää uudelleen siirrettäessä verkkopalvelua esimerkiksi palvelimesta toiseen. Ei-uudelleenkäytettävä osuus puolestaan sisältää sellaisia tietoja, kuten verkkopalvelun portti, IP-osoite ja palvelun polku, jotka voivat muuttua palvelun vaihtaessa sijaintiaan.



KUVIO 15. WSDL-dokumentin rakenne Adatian ym. (2001, 1007) mukaan.

Verkkopalvelun kuvaus on luonteeltaan funktionaalinen niiltä osin, kun se kuvaa sitä, kuinka verkkopalvelua kutsutaan ja mitä syötteitä ja tulosteita verkkopalvelu ymmärtää. Verkkopalvelun kuvaukseen kuuluu kuitenkin myös ei-funktionaalisia ominaisuuksia, jotka vastaavat siihen kysymykseen, miksi palvelun hakija ylipäätään haluaisi käyttää tiettyä verkkopalvelua. (Boubez ym. 2001, 315.)

Boubezin ym. (2001, 315-316) mukaan ei-funktionaalisia ominaisuuksia kuvaa kaksi erillistä standardia: WSEL (Web Service Endpoint Language) sekä WSFL (Web Service Flow Language). WSEL kuvaa sellaisia ominaisuuksia, jotka liittyvät tiettyyn verkkopalvelun tarjoajaan, kuten yhteyden laatua ja tietoturvaa. WSFL puolestaan kuvaa sellaista verkkopalveluiden joukkoa, jota voidaan käyttää kokonaisen liiketoimintaprosessin toteuttamisessa. WSFL edellyttää, että verkkopalvelujoukon yksittäisiä palveluita kutsutaan tietyssä järjestyksessä. (Boubez ym. 2001, 315-316.)

5.2.3 Palvelun hakeminen

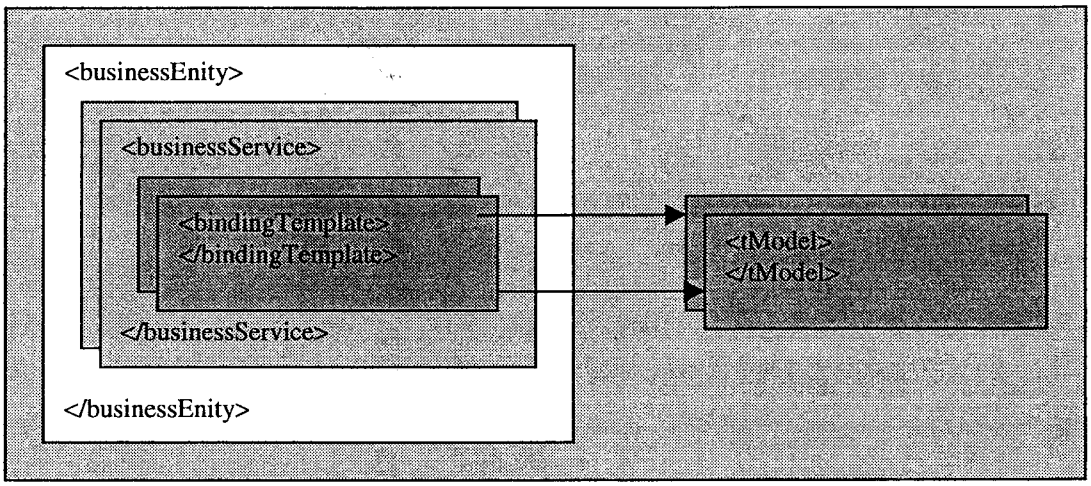
Verkkopalvelut ovat luonteeltaan dynaamisia, mikä edellyttää sitä, että niitä voidaan hakea tiettyjen kriteereiden perusteella sekä tehdä vertailuja valittujen ominaisuuksien perusteella. Erilaisten verkkopalvelujen hakemiseksi tarvitaan lisäksi jokin paikka, johon verkkopalveluita voidaan julkaista ja joka ylläpitää tietoa julkaistuista palveluista sekä siitä kuinka niitä voidaan kutsua. Tällaista paikkaa kutsutaan palvelurekisteriksi ja sitä voidaan verrata hajautettujen järjestelmien rekistereihin, jotka ylläpitävät tietoja järjestelmien eri osien sijainnista.

UDDI on organisaatio, joka on tehnyt määrittelyn verkkopalveluita ylläpitävästä palvelurekisteristä. Organisaation toiminnassa on mukana satoja yrityksiä, jotka voivat vaikuttaa standardin kehittämiseen. UDDI tukee W3C:n standardoimia ja verkkopalveluissa käytettäviä teknologioita, kuten HTTP:tä, XML:ää ja SOAP:ia sekä pyrkii olemaan alustariippumaton ja avoin arkkitehtuuri. (www.uddi.org.) Näiden seikkojen voidaan ennakoida edesauttavan kyseisen tekniikan muotoutumista

eräänlaiseksi de-facto –standardiksi verkkopalvelurekisterinä, koska verkkopalveluiden yhtenä periaatteena on olla helposti ja yleisesti kutsuttavissa.

Verkkopalveluiden tietoja säilytetään UDDI-rekisterissä XML-dokumenteissa, jotka kuvaavat rekisterin tarjoamia palveluita. Rekisteri koostuu liiketoiminta- ja teknologiamallista. *Liiketoimintamallissa* verkkopalvelun tiedot on jaettu kolmeen luokaan: valkoisiin, keltaisiin ja vihreisiin sivuihin. Valkoiset sivut sisältävät yleisiä tietoja verkkopalvelusta, jotka liittyvät palvelua tarjoavaan tahoon, kuten osoitetiedot. Keltaiset sivut sisältävät tietoa, jonka perusteella verkkopalvelu voidaan luokitella kuuluvaksi tiettyyn teollisuudenalaan. Vihreät sivut puolestaan sisältävät teknistä tietoa palvelusta sekä siitä, kuinka sitä voidaan kutsua. *Teknologiamallissa* määritellään tarkemmin se, kuinka verkkopalvelua kutsutaan. Vihreät sivut viittaavat täten teknologiamallin tietoihin. (Boubez ym. 2001, 395-397.)

KUVIOSSA 16 on kuvattuna UDDI:n XML-tietorakenne sekä rakenteiden väliset suhteet. XML-dokumentin businessEntity-elementti kuvaa yhden organisaation tietoja, eli se vastaa valkoisten sivujen tietoja. Toinen elementti, businessService, sisältää kuvauksen verkkopalvelun teknisistä ominaisuuksista sekä kaksi alielementtiä: bindingTemplate ja categoryBag. CategoryBag-elementti mahdollistaa luokittelevan tiedon liittämisen verkkopalveluun ja bindingTemplate-elementissä kuvataan verkkopalvelun kutsumispolku, joka voi olla URL-osoite tai jopa puhelinnumero. Se voi myös sisältää viittauksen tModel-elementtiin, joka kuvaa teknologiamallia. (Adatia ym. 2001, 1011.)



KUVIO 16. UDDI:n XML-tietorakenne Adatian ym. (2001, 1001) mukaan.

UDDI:n sisältämien palvelutietojen luokitteleminen erilaisiin ryhmiin auttaa tietojen hakemisessa sekä verkkopalveluiden löytämisessä. Keltaisilta sivuilta voidaan hakea esimerkiksi tietyn teollisuusalan verkkopalveluita sekä verrata niistä perittäviä maksuja. Tämän jälkeen verkkopalvelun toimittajan voidaan käyttää sitä, joka tarjoaa palvelua edullisimmin. Kilpailuttaminen voisi näin ollen tapahtua automaattisesti tiettyjen liiketoimintasääntöjen perusteella. Liiketoimintasäännöt tullaan todennäköisesti tulevaisuudessa toteuttamaan agenttiteknologian avulla, jolloin agentit voivat itsenäisesti hakea tietoa erilaisista verkkopalveluista sekä vertailla niiden ominaisuuksia (Boubez ym. 2001, 529.)

Adatian ym. (2001, 1012) mukaan UDDI-rekistereiden on tarkoitus muodostaa verkko, jolloin ne voivat vaihtaa tietoja keskenään. Rekisteriverkon toimintaperiaate vastaisi siten Internet-osoitteiden DNS (Domain Name System) palvelua, jolloin verkkopalvelun hakupyyntö välitettäisiin eteenpäin useille muille UDDI-rekistereille. Tällöin riittää, että verkkopalvelu julkaistaan yhteen julkiseen UDDI-rekisteriin, minkä jälkeen kaikki muutkin rekisterit voivat nähdä kyseisen palvelun. (Adatia ym. 2001, 1012.)

UDDI-rekisterit voivat olla paitsi julkisia myös yksityisiä, jolloin yksi organisaatio vastaa sen ylläpidosta. Boubez ym. (2001, 450-451) mainitsevat muutamia seikkoja miksi organisaatio voisi haluta yksityisen rekisterin. Joskus rekistereissä voi olla

arkaluontoisia palveluita, joita organisaatio ei halua kaikkien näkevän. Tällöin se voi yksityisen rekisterin avulla kontrolloida sisäänpääsyä, jolloin esimerkiksi vain kumppanit pääsevät käyttämään palveluita. Julkisessa rekisterissä voidaan kuitenkin julkaista tietoja yksityisestä rekisteristä, jolloin asiasta kiinnostuneet voivat käydä katsomassa kuinka sen käyttäjäksi pääsee. Toinen syy oman UDDI-rekisterin ylläpitoon voi olla verkkopalveluiden kuvaustietojen yhdenmukaisuus. Tällöin voidaan varmistua siitä, että kuvaukset noudattavat tiettyjä minimivaatimuksia ja mahdollistavat palvelukutsujen automaattisen generoinnin. (Boubez ym. 2001, 450-451.)

5.3 Verkkopalvelut ja liiketoimintaprosessit

Verkkopalveluarkkitehtuuri mahdollistaa palvelun julkaisemisen, hakemisen ja kutsumisen. SOAP:n, WSDL:n ja UDDI:n avulla voidaan kuitenkin rakentaa vain hyvin yksinkertaisia verkkopalveluita. Yksinkertaisilla verkkopalveluilla tarkoitetaan tässä yhteydessä sellaisia palveluita, jotka ovat suhteellisen lyhytkestoisia ja joiden toteuttamiseen osallistuu vain yksi osapuoli. Verkkopalvelut toimivat kuitenkin yleensä hajautetussa järjestelmässä, minkä lisäksi palveluiden tarjoamiseen voi osallistua useita eri osapuolia.

5.3.1 Työnkulku

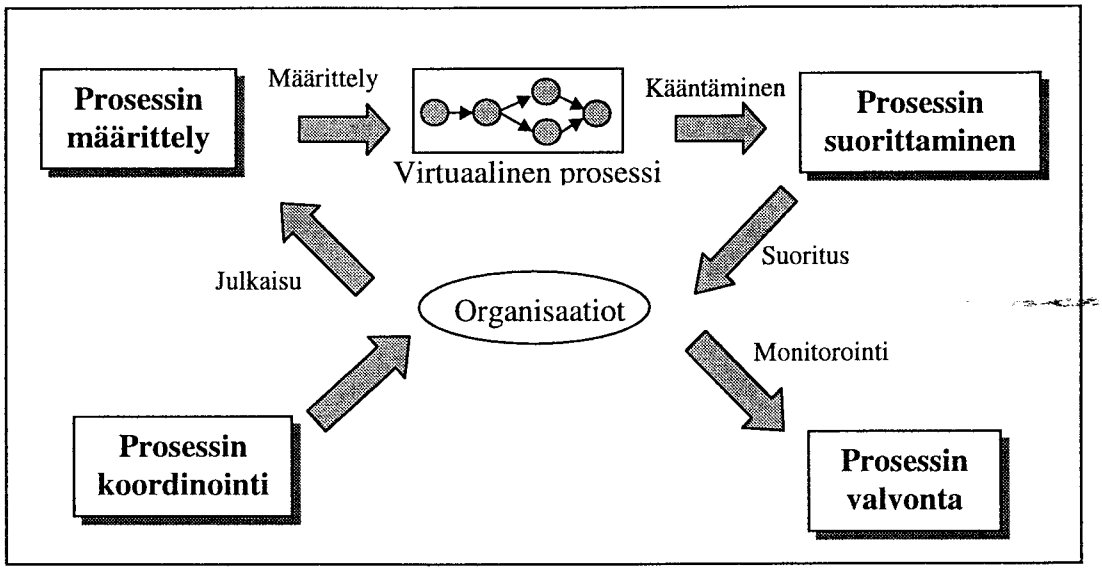
Liiketoimintaprosessi voidaan nähdä tehtävien joukkona, jossa useampi osapuoli osallistuu prosessin toteuttamiseen. Hollingsworth (1995, 6) käyttää tällaisesta liiketoimintaprosessin koordinoinnista nimeä työnkulku (workflow) ja se on hänen mukaansa toimintojen automatisointia, joissa dokumentteja, tehtäviä tai tietoja jaetaan eri osapuolten välillä.

Alonso ym. (1999, 132) ovat tutkineet organisaatioiden välistä liiketoimintaprosessien koordinointia ja työnkulun hallintaa virtuaalisen organisaation näkökulmasta. He ovat

kehittäneet WISE-mallin (Workflow based Interned Services), joka kuvaa organisaatioiden välisten prosessien määrittely-, suoritus- ja valvontatehtäviä. (Alonso 1999, 132.)

Virtuaalinen organisaatio on Alonson ym. (1999, 133-134) mukaan sellainen, jolla on virtuaalisia prosesseja. Virtuaalisilla prosesseilla he puolestaan tarkoittavat prosesseja, jotka läpäisevät useamman kuin yhden organisaation ja joista voidaan koota suurempia kokonaisuuksia. Esimerkkinä virtuaalisesta organisaatiosta ja prosessista mainitaan vakuutusyhtiö, jossa käsitellään kahta eri vahinkotyyppiä: varkauksia ja tulipaloja. Varkauksia koskevat vahingonkorvausarvioinnit suoritetaan vakuutusyhtiön sisällä. Tulipalojen tapauksessa vahingonarviointiprosessin suorittaakin ulkopuolinen organisaatio, jolloin vahingonarviointiprosessi voidaan nähdä virtuaalisena prosessina. (Alonso 1999, 133-134.)

Alonso ym. (1999, 133-134) pitävät haasteena sellaisen kokonaisarkkitehtuurin luomista, jonka avulla palveluita voidaan julkaista, hakea ja integroida organisaation omiin prosesseihin. Tämän lisäksi haasteena nähdään virtuaalisten prosessien hallinta ja valvonta. WISE-arkkitehtuuri on jaettu neljään eri pääkomponenttiin: prosessin määrittelyyn, suoritukseen, koordinointiin ja valvontaan. Komponenttien väliset suhteet on esitetty KUVIOSSA 17.



KUVIO 17. Virtuaalisen organisaatioon kuuluvat komponentit ja niiden väliset suhteet Alonson ym (1999, 135) mukaan.

Arkkitehtuurissa oletetaan, että uusia prosesseja voidaan luoda käyttämällä muiden organisaatioiden tarjoamia palveluita, jotka voivat itsessään jo olla prosesseja. Prosessien määrittely –komponentissa luodaan prosessin runko, joka määrittelee prosessin yleisen rakenteen. Määrittelykomponenttiin kuuluu myös palveluina tarjottavien prosessien julkaiseminen. Syntyneet prosessimäärittelyt käännetään tämän jälkeen sellaiseen muotoon, että niitä voidaan suorittaa työnkulun hallintajärjestelmässä. Prosesseja pitää lisäksi voida valvoa, jotta prosessin etenemistä ja siinä mahdollisesti ilmenneitä virheitä voidaan seurata. Prosesseissa sattuvat poikkeukset käsitellään koordinointikomponentissa, jonka kautta organisaatiot voivat kommunikoida toistensa kanssa. Koordinointi on tärkeää, koska virtuaalisen prosessin suorittamiseen voi osallistua useita eri organisaatioita. (Alonso ym. 1999, 135-137.)

5.3.2 Hajautetut prosessit

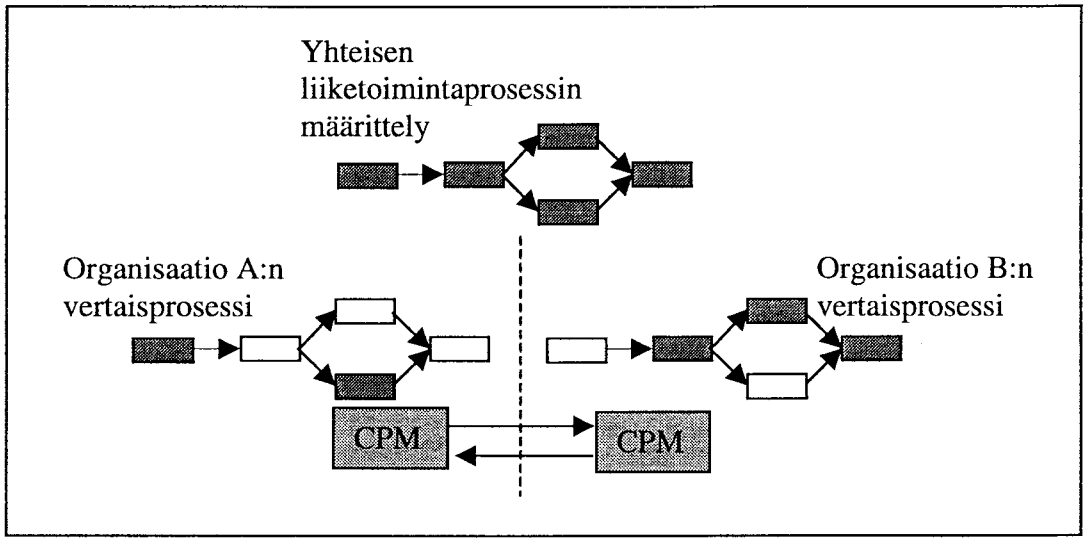
Chenin ym. (2001, 403-404) mukaan useamman organisaation välisen liiketoimintaprosessin integroiminen edellyttää hyvää työnkulun koordinointia. Koordinointia varten tarvitaan työnkulun hallintajärjestelmä, joka vastaa siitä, että liiketoimintaprosesseja suoritetaan oikeassa järjestyksessä ja oikeaan aikaan. (Chen ym. 2001, 403-404.)

Chen ja Hsu (2001, 253) toteavat, että organisaatioiden sisäinen prosessien hallinta eroaa huomattavasti organisaatioiden välisestä prosessien hallinnasta. Suurimpana syynä tähän on heidän mukaan se, että yhden organisaation on helppo sisäisesti päättää prosessien hallinnasta, kun taas organisaatioiden välistä prosessien hallintaa vaikeuttavat palomuurit, eturistiriidat ja prosessissa käytetyn datan suojelu ulkopuolisilta. Organisaatioiden pitäisikin keskitetyn prosessin hallinnan sijaan käyttää hajautettua prosessin hallintaa. (Chen ja Hsu 2001, 253.)

Hajautettu prosessin hallinnan keskeisenä osana toimii Chenin ja Hsun (2001, 253-254) mukaan yhteistoiminnallinen prosessin hallitsija CPM (Collaborative Process Manager). CPM toimii vertaisverkko-periaatteella, mikä tarkoittaa sitä, että kullakin

organisaatiolla on itsenäisesti toimiva CPM-palvelin. CPM-palvelimet jakavat keskenään julkisen prosessimääritelmän, mutta niillä voi myös olla prosessiin liittyvää tietoa ja aliprosesseja, jotka ovat yksityisiä. (Chenin ja Hsun 2001, 253-254.)

KUVIOSSA 18 on esitetty hajautetun prosessien hallinnan toimintaperiaate. Toiminnan pohjalla on yhteinen prosessimäärittely, joka muodostaa prosessin rungon. Organisaatiot A ja B osallistuvat kuitenkin vain prosessin tiettyihin vaiheisiin, minkä jälkeen suoritusvuoro siirretään toiselle. Suoritusvuoron koordinoinnista ja synkronoinnista vastaa kaksi CPM-palvelinta, jotka sijaitsevat sekä organisaatiossa A että organisaatiossa B.



KUVIO 18. Vertaisverkkoon perustuva yhteisten prosessien hallinta Chenin ja Hsun (2001, 254) mukaan.

Myös Benatallah ym. (2002, 297-298) pitävät vertaisverkkoon perustuvaa prosessien hallintaa erittäin houkuttelevana ratkaisuna ja perustelevat sitä verkkopalveluiden dynaamisella ja hajautuneella luonteella. Heidän mukaansa koosteisia verkkopalveluita voidaan hallita hajauttamalla prosessien koordinointi eri organisaatioille. (Benatallah ym. 2002, 297-298.)

5.3.3 Verkkopalveluiden transaktiot

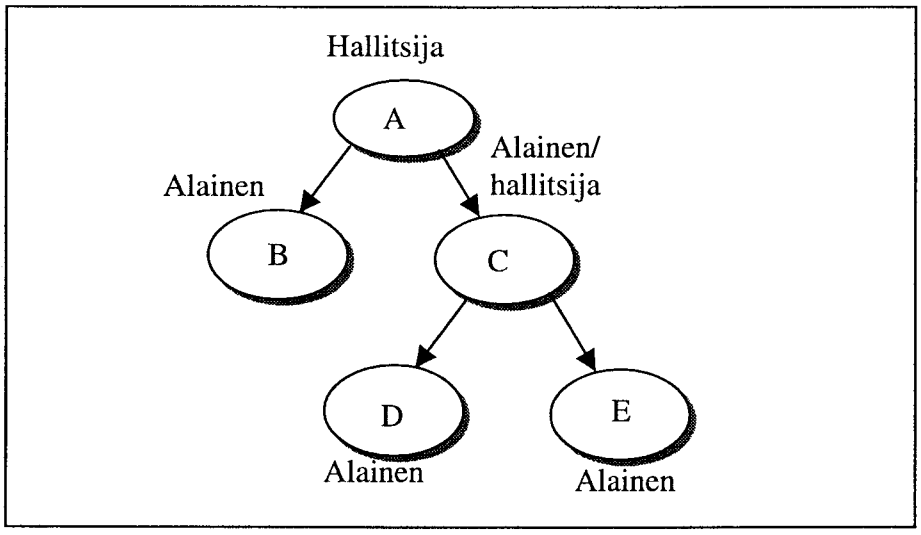
Verkkopalvelut toimivat yleensä hajautetussa järjestelmässä, minkä lisäksi palveluiden tarjoamiseen voi osallistua useita eri osapuolia. Eri osapuolten välisten järjestelmien koordinoinnin yksi tärkeä osa-alue on transaktiot. Romanovskyn (2001, 66) mukaan perinteiset ACID-määritykset voivat muodostua ongelmallisiksi tietyissä tilanteissa. Ongelmia voi esiintyä, jos käytössä on pitkiä transaktioita. Tällöin resurssit voivat olla pitkään lukittuina ja vapautua vasta sen jälkeen, kun transaktio on joko vahvistettu tai peruttu. Toinen ongelmakohta liittyy virhetilanteiden hallintaan, joka perustuu lähinnä transaktioiden peruuttamiseen tai vahvistetun tiedon palauttettavuuteen tietokantapalvelimen kaaduttua. ACID-määritykset toimivat kuitenkin huonosti tilanteissa, joissa transaktio on jäänyt epäyhtenäiseen tilaan ja joissa tarvittaisiin korjaavia toimenpiteitä. (Romanovsky 2001, 66.)

Yksinkertaiset transaktiot toimivat myös Krishnamoortyn ja Shanin (2000, 876) mukaan hyvin perinteisissä tietokantajärjestelmissä, joissa operaatiot ovat suhteellisen lyhytkestoisia ja joissa ACID-määritykset yleensä toteutuvat. Yksinkertaisten transaktioiden heikkoudet tulevat kuitenkin ilmi pitkäkestoisissa ja useita osapuolia koskevissa transaktioissa. (Krishnamoorty ja Shan 2000, 876.)

Potts ja Temel (2001, 3) pitävät perinteisiä ACID-transaktioita liian rajoittuneina, koska ne perustuvat yleensä koko transaktion peruuttamiseen virhetilanteen sattuessa. Pitkäkestoisissa transaktioissa osa näistä transaktioon osallistuneista resursseista voi kuitenkin olla saavuttamattomissa. Potts ja Temel nimittävätkin tällaisia pitkiä transaktioita liiketoimintatransaktioiksi ja niiden peruuttamiseen pitäisi heidän mukaansa soveltaa kompensoivaa lähestymistapaa. Tällöin yksittäiset lyhytkestoiset transaktiot noudattaisivat ACID-määrityksiä, mutta koko liiketoimintatransaktion mittakaavassa ACID-määrityksistä voitaisiin tinkiä kompensoivien toimenpiteiden avulla. (Potts ja Temel 2001, 3.)

OASIS (Organization for the Advancement of Structured Information Standards) – järjestön määrittelemä BTP (Business Transaction Protocol) on standardi, joka on luotu hajautettujen järjestelmien transaktion hallintaa varten. BTP:n tarkoituksena on mahdollistaa useamman itsenäisen organisaation välisen sovellusliikenteen koordinointi. Standardi käyttää tähän kaksivaiheista transaktion hallintaa, joka mahdollistaa kaikki-tai-ei-mitään transaktiot (atomic business transaction) tai transaktioiden osittaisen epäonnistumisen (cohesive business transaction). BTP mahdollistaa lisäksi kompensoivien transaktioiden käyttämisen poikkeusten sattuessa ja se on protokollariippumaton ja suunniteltu erityisesti uusia verkkopalveluprotokollia ja XML:ää silmällä pitäen. (Ceponkus ym. 2002, 11.)

Cepokus ym. (2002, 19) jakavat hajautettuun transaktioon osallistuvat osapuolet kahteen rooliin: hallitsijaan (superior) ja alamaiseen (inferior). Hallitsijoita ja alaisia voi olla useita ja ne voivat muodostaa puumaisia rakenteita KUVIOSSA 19 esitetyllä tavalla.



KUVIO 19. Liiketoimintatransaktion muodostama puu Ceponkusta ym. (2002, 29) mukaillen.

Edellä mainittu BTP-protokolla käyttää kaksivaiheista transaktion hallintaa. Tämä toimii Kitsuregawan ja Reddyn (1998, 408) mukaan siten, että hajautetun transaktion

hallitsija pyytää ensin alaisia valmistautumaan transaktion vahvistamiseen, minkä jälkeen hallitsija jää odottamaan niiden vastausta. Alaiset tarkastavat sen, voivatko ne vahvistaa omat paikalliset transaktionsa. Jokainen alainen lähettää tämän jälkeen hyväksymis- tai hylkäämisviestin transaktion hallitsijalle. Jos yksikin alainen jättää transaktion vahvistamatta transaktiota, niin hallitsija lähettää peruuttamisviestin kaikille alaisille, minkä jälkeen ne peruuttavat transaktion paikallisesti. Jos taas kaikki alaiset ovat lähettäneet transaktion hyväksymisviestin, niin hallitsija käskee alaisia vahvistamaan paikallisesti transaktion. (Kitsuregawa ja Reddy 1998, 408.)

Kuten edellisestä kuvauksesta havaitaan, kaksivaiheiseen transaktion hallintaan liittyy useita eri vaiheita ja paljon kommunikointia transaktioon osallistuvien osapuolten välillä. Adatian ym. (2001, 393-394) mukaan hajautetuissa transaktioissa onkin monia negatiivisia puolia, kuten epävarmat tietoliikenneyhteydet, manuaalista työtä vaativat korjaustehtävät, tehottomuus verrattuna paikallisesti suoritettaviin transaktioihin ja järjestelmien monimutkaisuus. (Adatian ym. 2001, 393-394.)

5.3.4 Verkkopalveluihin liittyviä standardeja

Verkkopalveluihin liittyviä standardeja on olemassa runsaasti, koska monet organisaatiot ovat liiketoimintalähtöisesti lähteneet kehittämään omia standardejaan. EbXML-standardin tavoitteena on Hofreiterin, Huemerin ja Klasin (2002, 8-9) mukaan luoda globaali elektroninen liiketoimintaympäristö, jossa organisaatiot voivat etsiä palveluita ja suorittaa liiketoimintaprosesseja. Se mahdollistaa myös prosessien ja tiedonvaihtoon liittyvien XML-dokumenttien määrittelyn, joita voidaan julkaista ja hakea ebXML-rekistereistä. EbXML tarjoaa myös reititys- ja tieturvapalveluita ja se on yhteensopiva SOAP:n kanssa. (Hofreiter, Huemer ja Klas 2002, 8-9.) Vaikka ebXML tarjoaa rekisteri ja varastointipalveluita, niin se ei sulje pois UDDI:n käyttötarvetta, vaan UDDI:a tullaan todennäköisesti käyttämään ebXML-rekistereiden paikallistamiseen (Kao 2001).

RosettaNet kehittää elektronisen liiketoiminnan standardeja informaatioteknologian sekä ja elektroniikkateollisuuden tarpeisiin. Se pyrkii määrittelemään prosesseja ja tietoa koskevia standardeja, joilla voidaan automatisoida tarjontaketjun eri osia. RosettaNetin prosesseja määritellään niin kutsutuilla PIP:llä (Partner Interface Process), jotka kuvaavat organisaatioiden välisiä rajapintoja. (Shim ja Sundaram 2001, 110-111; 113.) Chenin ja Hsun (2001, 255) mukaan PIP:t esittävät kuitenkin pelkästään määrittelyn siitä tiedosta, joka välitetään liiketoimintaprosessin eri osapuolten välillä. Se ei näin olleen ota kantaa itse liiketoimintaprosessin toteutukseen tai aliprosessien väliseen synkronointiin.

Allamaraju (2001, 587) mainitsee lisäksi muutaman muun verkkopalveluteknologian, jotka on tarkoitettu nimenomaan organisaatioiden väliseen tietojen vaihtoon. *BizTalk* on Microsoftin kehittämä sovelluskehys, joka on tarkoitettu elektronista liiketoimintaa käyttäville yrityksille. Se tosin vaatii Microsoftin oman BizTalk palvelimen ja Microsoftin määrittelemät XML-dokumentit. BizTalk-palvelin toimii palvelurekisterinä, jonka kautta yritykset voivat vaihtaa tietoja ja suorittaa liiketoimia. (Allamaraju 2001, 587.)

CommerceOne on organisaatio, joka tarjoaa rekisteripalveluita organisaatioiden väliseen tietojen vaihtoon. Tietojen vaihtamiseen käytetään xCBL:ää (XML Common Business Library), joka määrittelee erilaisia liiketoiminnassa käytettäviä dokumenttityyppejä. Nämä sisältävät mm. tilaukseen, tilauksenvahvistukseen, tarjouspyyntöön ja laskutukseen liittyviä XML-pohjaisia määrityksiä. (Allamaraju ym. 2001, 589.)

5.4 Yhteenveto

Verkkopalveluarkkitehtuuria voidaan käsitteellisesti verrata hajautettuihin komponenttiteknologioihin, kuten EJB:hen, jossa suljetut protokollat vain on korvattu yleisesti käytetyillä standardeilla. Sovellusten jatkuva monimutkaistuminen johtaa siihen, että monimutkaisuuden hallintaan täytyy käyttää yhä suurempi osa resursseista. Yksi keino vähentää monimutkaisuutta on abstraktio. Tulevaisuudessa abstrahointia

voidaan periaatteessa jatkaa siten, että kokonaisia järjestelmiä aletaan käsittelemään komponentteina. Verkkopalvelut mahdollistavat rajapintakuvauksensa ansiosta palveluiden hakemisen ja dynaamisen sidonnan, jolloin organisaatiot voivat integroida liiketoimintaprosessejaan entistä nopeammin.

Verkkopalveluarkkitehtuurin perusmalliin kuuluu kolme osapuolta: palvelun hakija, palvelun tarjoaja ja palvelun välittäjä. Palvelun välittäjä ylläpitää verkkopalveluiden rajapintakuvauksia, jotka mahdollistavat palveluiden hakemisen ja kutsumisen. Palvelun hakija voi etsiä sopivaa verkkopalvelua tämän rajapintakuvauksen avulla ja sitoa oman sovelluksensa käyttämään kyseistä palvelua. Palvelua kutsuttaessa pyyntö välittyy palvelun tarjoajalle, joka lopulta palvelee kutsun tehnyttä osapuolta.

Verkkopalveluarkkitehtuurin kolmeksi perusstandardiksi ovat muovautumassa XML:ään perustuvat SOAP, WSDL ja UDDI. SOAP on HTTP-protokollan päällä toimiva etäkutsuprotokolla, jolla palveluita voidaan hakea ja kutsua. WSDL puolestaan on verkkopalveluiden rajapintoja kuvaava kieli, ja se erottaa palvelun kuvauksen sen varsinaisesta toteutuksesta. UDDI:n tehtävänä on toimia palvelukuvauksia ylläpitävänä ja välittävänä rekisterinä.

Jotta verkkopalveluita voitaisiin käyttää oikeiden liiketoimintaprosessien toteuttamiseen, tarvitaan kuitenkin lisäominaisuuksia perusteknologioiden päälle. Yksi tärkeimmistä on prosessien hallintaan liittyvä työnkulku, joka ohjaa liiketoimintaprosessien toimintaa ja koordinoi prosessien välistä kommunikointia. Verkkopalveluiden perusarkkitehtuurista puuttuu lisäksi prosessien määrittelyyn liittyviä komponentteja. Tällaisia liiketoimintaprosessien määrittelyä tukevia standardeja on olemassa lukuisia, mutta teknologiat ovat vielä osittain hajanaisia ja niiden integrointi voi olla työlästä.

6 POHDINTA

Tekniikan kehitys on mahdollistanut manuaalisten ja rutiinitehtävien automatisoinnin. Näin on tapahtunut myös elektronisessa liiketoiminnassa, jossa monet prosessit on siirretty verkon kautta suoritettaviksi. Elektroninen liiketoiminta tulee kehittymään yhä tiiviimmäksi osaksi organisaatioiden prosesseja, mikä kasvattaa integrointitarvetta sekä organisaation sisäisissä että organisaation välisissä järjestelmissä.

Organisaation sisäisten järjestelmien integroiminen verkkokauppaan voidaan katsoa edustavan B2C-mallin mukaista toimintaa. B2C-mallin mukaisia ratkaisuja voidaan kehittää yleisimmillä komponenttiteknologioilla, jotka tukevat liiketoiminta-komponenttien mallintamista liiketoimintaprosessilähtöisesti. Yksi tällainen komponenttiteknologia on EJB, jonka etuna on valmis komponenttimalli, hyvä skaalautuvuus sekä binääriprotokollan mukanaan tuoma nopeus. Edustajärjestelmien integraatio operationaalisiin järjestelmiin kuuluu elektronisen kaupan vaiheeseen ja tällöin tärkeitä huomioitavia seikkoja ovat järjestelmien suorituskyky ja asiakkaan kokema laatu.

Yksittäinen komponenttimalli rajoittuu kuitenkin lähinnä organisaatioiden sisäisten järjestelmien integrointiin edustajärjestelmien kanssa, koska organisaatiot ovat haluttomia paljastamaan sovellustensa rajapintoja ulkopuolisille. Tämän lisäksi komponenttimalli ei itsessään tarjoa elektronisen liiketoiminnan kannalta keskeisiä ominaisuuksia, kuten määrämuotoisten viestien välitystä ja ylemmällä tasolla tapahtuvaa liiketoimintaprosessien koordinointia. Nämä ominaisuudet pitääkin rakentaa itse uusien liiketoimintakumppanien kanssa, mikä voi olla työlästä ja hidasta. Tämä aiheuttaa myös sen, että järjestelmät ovat suhteellisen staattisia ja liiketoimintasuhteet kiinteitä ja pitkäaikaisia.

Elektronisen liiketoiminnan kehityksen yhtenä tasona on verkostotalous, jossa organisaatioiden väliset liiketoimintaprosessit ovat dynaamisesti konfiguroitavissa. Organisaatioiden välinen tiedonsiirto vaatiikin uusia muotoja, jos tavoitteena on saada kustannusetuja ja vastata kovenevaan kilpailuun. Yhtenä edellytyksenä dynaamisille prosesseille on yhteisten standardien käyttäminen, mikä helpottaa erilaisten järjestelmien välistä kommunikointia. Verkkopalvelustandardit ovatkin yksi merkittävä askel eteenpäin siirryttäessä kohti verkostomaisempaa toimintatapaa.

Verkkopalveluiden ideana on mahdollistaa organisaatioiden palveluiden käyttäminen julkisesti saatavilla olevan rajapinnan kautta. Verkkopalvelu voidaan rinnastaa komponenttiin, jonka abstraktiotasoa vain on hieman nostettu. Tällöin erilliset järjestelmät voidaan nähdä komponentteina, jotka tarjoavat palveluita standardiprotokollien ja rajapintojen kautta. TAULUKKOON 3 on koottu yhteenvetona Herzumin ja Simsin (1999) esittämä komponenttien karkeustasoluokittelu sekä näiden luokkien vastaavuus elektronisen liiketoiminnan sovelluksen eri osien kanssa.

TAULUKKO 3. Herzumin ja Simsin (1999) komponenttien karkeustasot ja niiden vastaavuus elektronisen liiketoiminnan sovellusten osien kanssa.

Komponentin tyyppi	E-liiketoiminnan sovelluksen osa
Hajautettu komponentti	Toiminto
Liiketoimintakomponentti	Liiketoimintaprosessi
Liiketoimintakomponenttien järjestelmä	Elektronisen liiketoiminnan sovellus, B2C
Systeemikomponenttien federaatio	Verkkopalvelut, B2B

TAULUKOSSA 3 kuvatusta komponenttien karkeustasoluokittelusta havaitaan, että kolme ensimmäistä komponenttiluokkaa (hajautettu-, liiketoiminta- ja systeemikomponentti) kuuluvat yhden organisaation sisäiseen järjestelmään. Tämä on se raja, jossa EJB-komponenttiteknologiaa voidaan helposti vielä hyödyntää. Jotta myös erilliset järjestelmät saataisiin näkymään komponentteina ja niitä voitaisiin kutsua järkevästi ilman suurta integrointityötä, tarvitaan verkkopalveluita. Tämä edellyttää sitä, että verkkopalvelun tarjoaja erotetaan palvelun rajapinnasta. Järjestelmistä tulee tällöin

avoimia, mikä mahdollistaa verkostomaisen toiminnan ja yksinkertaistaa organisaatioiden välisten prosessien integrointia.

Perusverkkopalveluteknologiat ovat SOAP, WSDL ja UDDI. Nämä tarjoavat yleisesti hyväksyttyihin standardeihin perustuvan verkkopalveluarkkitehtuurin, joka mahdollistaa palveluiden hakemisen, julkaisemisen ja kutsumisen. Niiden hyvänä puolena on XML-pohjaisuus, joka on käyttöjärjestelmäriippumaton ja rakenteellinen tiedon esitystapa. Protokollien tekstipohjaisuus aiheuttaa kuitenkin suorituskykyongelmia, eikä verkkopalveluita näin ollen kannata käyttää aikakriittisten sovellusten pohjana.

Verkkopalveluiden todellinen hyöty saavutetaan B2B-sovelluksissa, joissa verkkopalveluiden dynaamista luonnetta voidaan hyödyntää. Verkostomainen palveluiden käyttö mahdollistaa organisaatioiden välisen prosessien integraation, jolloin voidaan saavuttaa elektronisen verkostotalouden vaihe (Järvelä ym. 2000, 8). Verkostotalouden toteutumisen mahdollistaa kolme tekijää, joiden suhde verkkopalveluihin näkyy TAULUKOSSA 4.

TAULUKKO 4. Elektronisen verkostotalouden tekijöiden suhde verkkopalveluun Järvelää ym. (2000, 8-9) mukaillen.

Tekijä	Elektroninen verkostotalous	Verkkopalvelu
Prosessit	Osittain jaetut ja ulkoistetut	Koosteisuus, verkottuneisuus
Teknologia	Protokollat ja standardit	XML, HTTP, WSDL, UDDI
Asiakassuhde	Asiakkaan hallitsema	Dynaamisuus, palvelun haku

TAULUKKOON 4 on koottu verkkopalvelun suhde niihin tekijöihin, jotka ovat tunnusomaisia verkostomaisessa palveluiden integroinnissa. Verkkopalveluiden *koosteisuus* ja *verkottuneisuus* mahdollistaa prosessien ulkoistamisen kolmansille osapuolille. Koostamisella tarkoitetaan tässä sitä, että verkkopalvelu voi kutsua toisia verkkopalveluita toteuttaakseen asiakkaan tekemän pyynnön. XML, HTTP, WSDL, UDDI ja muut tekniset *standardit* toteuttavat puolestaan teknologisen ehdon, jonka mukaan elektronisessa verkostotaloudessa käytetään standardeja kommunikoinnissa ja

tiedon siirtämisessä. Verkkopalveluiden *dynaamisuus* siirtää neuvotteluvoimaa asiakkaalle, joka voi valita palveluista ne, jotka parhaiten vastaavat niitä kriteereitä, joita hän on palveluille asettanut.

Verkkopalveluarkkitehtuuri vaatii kuitenkin laajennuksia, jotta sitä voitaisiin käyttää oikeissa liiketoimintaprosesseissa. Liiketoimintaprosessien yhtenä tärkeänä kulmakivenä on työnkulku, joka määrittelee sen kuinka liiketoimintaprosessia koordinoidaan ja valvotaan. Ilman liiketoimintaprosessin koordinointia voidaan toteuttaa vain hyvin yksinkertaisia verkkopalveluita. Koosteisten ja useita osapuolia käsittävien pitkien liiketoimintaprosessien toteuttamiseen tarvitaan prosessin työnkulun koordinointia. Yhtenä suurimpana ongelmana koordinoinnissa on puolueettoman koordinaattorin aikaansaaminen, koska organisaatiot ovat yleensä haluttomia paljastamaan omissa prosesseissaan liikkuvaa tietoa.

Koosteisten verkkopalveluiden käyttäminen merkitsee useiden organisaatioiden osallistumista liiketoimintaprosessien toteuttamiseen, jolloin toiminta tapahtuu hajautetussa ympäristössä. Hajautettujen järjestelmien välisiin transaktioihin käytetään yleensä kaksivaiheista transaktion vahvistamista. Tämä aiheuttaa transaktioille suorituskyky- ja luotettavuusongelmia, mikä tekee verkkopalveluiden käyttämisestä liiketoimintakriittisissä sovelluksissa riskialtista. Myös kompensoivat transaktiot voivat aiheuttaa ongelmia, koska täysin automaattisessa prosessissa on vaikea etukäteen määritellä sellainen kompensaatio, joka aina vastaa menetettyä hyötyä.

Vaikka verkkopalvelut vaikuttavatkin erinomaiselta keinolta automatisoida prosesseja ja leikata integrointikustannuksia, niin niiden käyttäminen tulee todennäköisesti aluksi olemaan vaatimatonta. Suurimpana syynä tälle on teknologian kypsymättömyys ja lukuisat verkkopalveluita lähellä olevat standardit, jotka vaikeuttavat kokonaisvaltaisen verkkopalveluarkkitehtuurin luomista. Kokonaisarkkitehtuuri on tärkeä, koska muuten on vaarana, että epäyhtenäiset standardit tekevät verkkopalveluiden käytöstä liian monimutkaista ja työlästä. On myös mahdollista, että mitään kokonaisvaltaista arkkitehtuuria ei synny, vaan organisaatiot alkavat käyttämään teknologioita, jotka ovat tietyiltä osin rajoittuneita, mutta ratkaisevat hyvin jonkin yksittäisen ongelman.

7 YHTEENVETO

Tutkielman tavoitteena oli tarkastella liiketoimintaprosessien toteuttamista Enterprise JavaBeans-komponenttimallin ja verkkopalveluiden avulla. Liiketoimintaprosessien nähtiin kehittyvän elektronisen liiketoiminnan tasojen mukaisesti siten, että staattisista prosesseista tultaisiin siirtymään kohti yhä dynaamisempia prosesseja. Dynaamisuuden lisääntymisen oletettiin kuitenkin aiheuttavan järjestelmien monimutkaistumista ja edellyttävän komponenteilta entistä abstraktimpaa luonnetta.

Tutkimusmenetelmänä käytettiin käsitteellisteoreettista tutkimusta, jossa kirjallisuuteen pohjautuen tarkasteltiin liiketoimintaprosessien toteuttamista sekä EJB:n että verkkopalveluiden avulla. Kirjallisuuden pohjana käytettiin komponenttiteknologiaan liittyviä kirjoja sekä ajankohtaisia artikkeleita, joilla pyrittiin luomana aiheeseen tuore näkökulma.

Tutkielmassa saatiin seuraavanalaisia tuloksia. EJB-komponenttimallin havaittiin tukevan erityyppisten liiketoimintaprosessien kehittämistä hyvin, koska se sisälsi valmiit komponentit sekä tietokohteiden, prosessien että viestinvälityksen toteuttamiseksi. Tämän lisäksi EJB 2.0 –standardin todettiin sisältävän useita komponenttimallin tehokkuuteen liittyviä uudistuksia. EJB-komponenttimallin todettiin kuitenkin tukevan huonosti organisaatioiden välisten liiketoimintaprosessien toteuttamista, koska se ei itsessään sisältänyt prosessien määrittelyyn ja koordinointiin liittyviä valmiita komponentteja.

Yksi tutkielman mielenkiintoisimpia tuloksia oli komponenttiteknologian rinnastaminen verkkopalveluihin, mikä onnistui komponenttikäsitteen abstraktiotasoa nostamalla. Tällöin erilliset järjestelmät voitiin nähdä komponentteina, jotka tarjoavat palveluita standardiprotokollien ja rajapintojen kautta. Verkkopalveluiden avulla voidaan ehkä tulevaisuudessa toteuttaa dynaamisia liiketoimintaprosesseja. Verkkopalveluiden todettiin kuitenkin vielä olevan kypsymättömiä ja edellyttävän liiketoimintaprosessien hallintaa tukevien mekanismien mukaan ottamista arkkitehtuuriin.

Tulokset eivät merkittävästi poikkea aikaisemmin tehdyistä tutkimuksista. Tutkielmassa on kuitenkin pyritty tuomaan esille uusia näkökulmia komponenttiteknologian käytön mahdollisuuksista liiketoimintaprosessien toteuttamisessa. Tutkielma ei esitä kaiken kattavaa näkemystä komponenttiteknologiasta eikä käsittele kaikkia liiketoimintaprosesseihin liittyviä tekijöitä. Tarkastelun rajoittuminen EJB:hen komponenttiteknologian osalta vähentää hieman tulosten yleistettävyyttä. Toisaalta se mahdollisti tarkemman perehtymisen yhden komponenttimallin erityispiirteisiin.

Jatkotutkimuksen kannalta mielenkiintoisia aiheita voisivat olla verkkopalveluiden tietoturvaan ja luotettavuuteen liittyvien tekijöiden tarkastelu. Toisena jatkotutkimusaiheena voisi olla verkkopalveluteknologioita lähellä olevien standardien tarkastelu.

LÄHDELUETTELO

Adatia, R., Arni, F., Berry, C. A., Gabhart, K., Griffin, J., Juric, M., Lott, J., McAllister, T., Mulder, A., Nagarajan, N., O'Connor, D., Osborne, T., Sarang, P. G., Tost, A. & Young, D. 2001. Professional EJB. Wrox Press Ltd, Birmingham, UK.

Allamaraju, S., Ashri, R., Darby, C., Flenner, R., Karsjens, T., Kerzner, M., Krotov, A., Linde, A., MacIntosh, J., McGovern, J., Mirchandani, T., Plaster, B., Reamy, D., Sarang, P. G. & Writz, D. 2001. Professional Java E-Commerce. Wrox Press Ltd, Birmingham, UK.

Alonso, G., Fieldler, U., Hagen, C., Lazcano, A., Schuldt, H. & Weiler, N. 1999. WISE: Business to Business E-Commerce. RIDE-VE '99, Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises, 1999, 132-139. Viitattu [19.5.2002]. Saatavana www-osoitteessa <http://www.ieeexplore.ieee.org/iel4/6091/16312/00758645.pdf?isNumber=16312&prod=CNF&arnumber=758645&arSt=132&ared=139&arAuthor=Alonso%2C+G.%3B+Fiedler%2C+U.%3B+Hagen%2C+C.%3B+Lazcano%2C+A.%3B+Schuldt%2C+H.%3B+Weiler%2C+N.>.

Ambler, S., Jewell, T. & Roman, E. 2002. Mastering Enterprise JavaBeans. Second Edition. Wiley Computing Publishing. John Wiley & Sons, Inc. New York.

Amor, D. 2000. The E-business (r)evolution. Prentice Hall, Upper Saddle River, NJ.

Benatallah, B., Dumas, M., Ngu, A. H. H. & Sheng, Q. Z. 2002. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. 18th International Conference on Data Engineering, 2002, 297-308. [Viitattu 13.6.2002]. Saatavana www-osoitteessa <http://fp.ieeexplore.ieee.org/iel5/7807/21451/00994738.pdf?isNumber=21451&prod=CNF&arnumber=994738&arSt=297&ared=308&arAuthor=Benatallah%2C+B.%3B+Dumas%2C+M.%3B+Sheng%2C+Q.Z.%3B+Ngu%2C+A.H.H.>.

Björn, H., Juneja, E., Kalilainen, T., Karhusaari, W. Nylander, T. & Rasimus, T. 2000. Digitaalinen tarjontaketju. WSOY, Juva.

Boubez, T., Daniels, G., Davis, D., Graham, S., Nakamura, Y., Neyama, R. & Simeonov, S. 2001. Building Web Services with Java; Making Sense of XML, SOAP, WSDL and UDDI. Sams Publishing, Indianapolis, Indiana, USA.

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., Winer, D. 2000. Simple Object Access Protocol (SOAP) 1.1, 8.5.2000. W3C, World Wide Web Consortium. [Viitattu 1.1.2002]. Saatavana www-osoitteessa <http://www.w3.org/TR/SOAP/>.

Brown, A. W. 2000. Large-scale Component-based Development. Prentice-Hall, Upper Saddle River, NJ.

Brugali, D., Fayad, M. E. & Hamu, D. S. 2000. Enterprise Frameworks Characteristics, Criteria and Challenges. Communications of the ACM, vol. 43 num. 10, 39-46. [Viitattu 23.10.2001]. Saatavana www-muodossa <http://delivery.acm.org/10.1145/360000/352200/p39-fayad.pdf?CFID=353822&CFTOKEN=64768529>.

Bussler, C. 2002. Modeling and Executing Semantic B2B Integration. Proceedings of the 12th International Workshop on Research Issues in Data Engineering: Engineering e-Commerce/e-Business Systems (RIDE'02), 69-74. [Viitattu 8.5.2002]. Saatavana www-osoitteessa <http://www.ieeexplore.ieee.org/iel5/7808/21452/00995100.pdf?isNumber=21452&prod=CNF&arnumber=995100&arSt=69&ared=74&arAuthor=Bussler%2C+C.>.

Casati, F. & Shan, M.-C. 2001. Dynamic and adaptive composition of e-services. Information Systems vol. 26, 143-163. Elsevier Science Ltd. [Viitattu 7.11.2001]. Saatavana www-osoitteessa <http://www.sciencedirect.com/web-editions?_ob=Mimg&_imagekey=B6V0G-4379VVY-4-1&_udi=B6V0G-4379VVY-4&_cdi=5646&_orig

=browse&_coverDate=05%2F31%2F2001&_sk=999739996&_acct=C000025142&_version=1&_userid=508775&md5=84a62a8feb69c454338028e4bf489222&ie=f.pdf>.

Cattell, R. G. G. & Inscore, J. 2001. J2EE Technology in Practice: Building Business Applications with the Java 2 Platform, Enterprise Edition. Addison-Wesley, UK.

Ceponkus, A., Dalal, S., Fletcher, T., Furniss, P., Green, A. & Pope, B. 2002. Business Transaction Protocol. An OASIS Committee Specification. Version 1.0 [0.95]. Viitattu [23.4.2002]. Saatavana www-osoitteessa <http://www.oasis-open.org/committees/business-transactions/documents/2002-04-03.BTP_draft_0.9.5.pdf>.

Chen, Q. & Hsu, M. 2001. Inter-Enterprise Collaborative Business Process Management. Data Engineering, 2001, Proceedings. 17th International conference on Data Engineering, 253-260. [Viitattu 21.4.2002]. Saatavana www-osoitteessa <http://ieeexplore.ieee.org/iel5/7304/19758/00914836.pdf?isNumber=19758&prod=CNF&arnumber=914836&arSt=253&ared=260&arAuthor=Qiming+Chen%3B+Meichun+Hsu>.

Chen, X., Miao, C., Shi, M., Zhang, Y. & Zhuang, Z. 2001. Workflow Interoperability – Enabling E-Business. The Sixth International Conference on Computer Supported Cooperative Work in Design, 2001, 403-408. [Viitattu 5.5.2002]. Saatavana www-osoitteessa < http://www.ieeexplore.ieee.org/iel5/7500/20395/00942294.pdf?isNumber=20395&prod=CNF&arnumber=942294&arSt=403&ared=408&arAuthor=Yan+Zhang%3B+Meilin+Shi%3B+Chunyu+Miao%3B+Zixin+Zhuang%3B+Xinxiang+Chen>.

Christensen, E., Curbera, F., Meredith, G. & Weerawarana, S. 2001. Web Services Description Language (WSDL) 1.1, 15.03.2001. W3C, World Wide Web Consortium. [Viitattu 1.1.2002]. Saatavana www-osoitteessa <http://www.w3.org/TR/wsdl>.

Crnkovic, I. & Larsson, M. 2000. A case study: demands on component-based development. Proceedings of the 22nd international conference on software engineering, 2000, Limerick, Ireland, 23-31. ACM Press, New York, NY, USA. [Viitattu 9.2.2002].

Saatavana www-osoitteessa <http://delivery.acm.org/10.1145/340000/337185/p23-crnkovic.pdf?key1=337185&key2=0922823101&coll=portal&dl=ACM&CFID=15185 45&CFTOKEN=40225712>.

Dahlén, T. 2001. Implementing Data Access Objects in an EJB app. Java Report, July 2001. 101communications LLC, UK.

Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. & Samarati, P. 2001. Fine Grained Access Control for SOAP E-Services. International World Wide Web Conference. The tenth international World Wide Web conference on World Wide Web, 2001, Hong Kong, Hong Kong. ACM Press, New York, USA. [Viitattu 1.1.2002]. Saatavana www-osoitteessa <http://delivery.acm.org/10.1145/380000/372152/p504-damiani.pdf?key1=372152&key2=7050989001&coll=portal&dl=ACM&CFID=112398 2&CFTOKEN=35945039>.

Daum, B. & Scheller, M. 2000. Success with Electronic Business – Design, Architecture and Technology of Electronic Business Systems. Addison-Wesley, Great Britain.

DeMichiel, L. G, Krishnan, S. & Yalçinalp, Ü. L. 2001. Enterprise JavaBeans Specification Version 2.0. August 2001, final release. Sun Microsystems, Palo Alto, California, U.S.A.

Emmerich, W., Finkelstein, A. & Piccinelli, G. 2001. Mapping Service Components to EJB Business Components. Enterprise Distributed Object Computing Conference, 2001. Proceedings. Fifth IEEE International, 2001, 169 – 173. [Viitattu 12.1.2002]. Saatavana www-osoitteessa <http://ieeexplore.ieee.org/iel5/7549/20561/00950434.pdf?isNumber= 20561>.

Gray, J. & Reuter, A. 1993. Transaction Processing: Concepts and Techniques. Morgan Kaufmann Publishers, Inc., San Mateo, CA, United States of America.

Hammer, M. & Champy, J. 1993. Reengineering the corporation. A manifesto for business revolution. HarperCollins publisher, HarperBusiness, New York.

Herzum, P. & Sims, O. 1999. Business Component Factory. A Comprehensive Overview of Component-Based Development for the Enterprise. John Wiley & Sons, Inc., The United States of America.

Heuvel, W-J., Papazoglou, M. P & Yang, J. 2002. Tackling the Challenges of Service Composition in E-Marketplaces. Proceedings of the 12th International Workshop on Research Issues in Data Engineering: Engineering e-Commerce/e-Business Systems RIDE-2EC 2002, 125-133. [Viitattu 1.6.2002]. Saatavana www-osoitteessa < http://www.ieeexplore.ieee.org/iel5/7808/21452/00995106.pdf?isNumber=21452&prod=CNF&arnumber=995106&arSt=125&ared=133&arAuthor=Jian+Yang%3B+Papazoglou%2C+M.P.%3B+van+den+Heuvel%2C+W.-J.>.

Hofreiter, B., Huemer, C. & Klas, W. 2002. EbXML: Status, Research Issues, and Obstacles. Proceedings of the 12th International Workshop on Research Issues in Data Engineering: Engineering e-Commerce/e-Business Systems RIDE-2EC 2002, 7-16. [Viitattu 13.6.2002]. Saatavana www-osoitteessa <http://www.ieeexplore.ieee.org/iel5/7808/21452/00995093.pdf?isNumber=21452&prod=CNF&arnumber=995093&arSt=7&ared=16&arAuthor=Hofreiter%2C+B.%3B+Huemer%2C+C.%3B+Klas%2C+W.>.

Hollingsworth, D. 1995. The Workflow Reference Model. Document Number TC00-1003. Document Status – Issue 1.1. Workflow Management Coalition Specification, UK. Viitattu [15.5.2002]. Saatavana www-osoitteessa <http://www.wfmc.org/standards/docs/tc003v11.pdf>.

Hopkins, J. 2000. Component Primer. Communications of the ACM, vol. 43 no. 10, 27-30. ACM Press, New York, NY, USA. [Viitattu 23.10.2001]. Saatavana www-muodossa <http://delivery.acm.org/10.1145/360000/352198/p27-hopkins.pdf?CFID=353786&CFTOKEN=55892137>.

Järvelä, P., Kallio, J., Laine, J., Loikkanen, J., Raijas, A., Raulas, M. Saarinen, L. M. J., Saarinen, T., Tinnilä, M., Tuunainen, V. K., Vepsäläinen, A. P. J. & Öörni, A. 2000. Elektronisesta kaupasta eLiiketoimintaan. Digitaalisen median raportti 1/2000. Paino-Center Oy, Sipoo.

Kalakota, R. & Whinston, A. B. 1996. Frontiers of Electronic Commerce. Addison-Wesley Publishing Company, Reading, Massachusettes.

Kao, J. 2001. Developer's Guide to Building XML-based Web Services with the Java 2 Platform, Enterprise Edition (J2EE). The Server Side, June 2001. [Viitattu 18.1.2002]. Saatavana www-osoitteessa <http://www.theserverside.com/resources/articles/WebServices-Dev-Guide/article.html>.

Kitsuregawa, M. & Reddy, P. K. 1998. Reducing the blocking in two-phase commit protocol employing backup sites. Proceedings, 3rd IFCIS Internationa Conference on Cooperative Information Systems, 406-415. Viitattu [1.5.2002]. Saatavana www-osoitteessa <http://www.ieeexplore.ieee.org/iel4/5723/15313/00706315.pdf?isNumber=15313&prod=CNF&arnumber=706315&arSt=406&ared=415&arAuthor=Reddy%2C+P.K.%3B+Kitsuregawa%2C+M.>.

Kobryn, C. 2000. Modeling Components and Frameworks with UML. Communications of the ACM, vol. 43 num.10, 31-38. ACM Press, New York, NY, USA. [Viitattu 23.10.2001]. Saatavana www-muodossa <http://delivery.acm.org/10.1145/360000/352199/p31-kobryn.pdf?CFID=353844&CFTOKEN=49619973>.

Krishnamoorty, V. & Shan, M-C. 2000. Virtual Transaction Model to Support Workflow Applications. Symposium on Applied Computing. Proceedings of the 2000 ACM Symposium on Applied Computing, 876-881, Como, Italy. ACM Press, New York, NY, USA. [Viitattu 1.5.2002]. Saatavana www-osoitteessa <http://delivery.acm.org/10.1145/340000/338581/p876-krishnamoorthy.pdf?key1=338581&key2=2360620201&coll=portal&dl=ACM&CFID=2416222&CFTOKEN=9483961>.

Matena, V. & Stearns, B. 2000. Applying Enterprise JavaBeans. Component-Based Development for the J2EE Platform. Addison-Wesley, Boston.

Matsuda, S., Yokoyama, K. & Yoshida, E. 2002. Requirements for Open Service Collaboration among Web Services. Proceedings, 2002 Symposium on Applications and the Internet (SAINT) Workshops, 214-217. [Viitattu 11.5.2002]. Saatavana www-osoitteessa <http://www.ieeexplore.ieee.org/iel5/7803/21447/00994572.pdf?isNumber=21447&prod=CNF&arnumber=994572&arSt=214&ared=217&arAuthor=Yokoyama%2C+K.%3B+Yoshida%2C+E.%3B+Matsuda%2C+S.>.

May, P. 2000. The Business of Ecommerce, from Corporate Strategy to Technology. Cambridge university press, UK.

Potts, M. & Temel, S. 2001. Business Transactions in Workflow and Business Process Management. OASIS Business Transactions Technical Committee, Workflow sub-committee, 7.12.2001. [Viitattu 13.6.2002]. Saatavana www-osoitteessa <http://www.oasis-open.org/committees/business-transactions/documents/2001-07-12.BTPModelForWF2.doc>.

Pour, G. 2000. Enterprise JavaBeans 101: Server-Side Components. Software Development Magazine [viitattu 23.10.2001]. Saatavana www-muodossa <http://www.sdmagazine.com/articles/2000/0004/0004b/0004b.htm>.

Ran, A. 1999. Software isn't built from Lego blocks. Proceedings of the fifth symposium on software reusability, 164-169, 1999, Los Angeles, California, UnitedStates. ACM Press, New York, NY, USA. [Viitattu 9.2.2002]. Saatavana www-osoitteessa < http://delivery.acm.org/10.1145/310000/303079/p164-ran.pdf?key1=303079&key2=7093823101&coll=portal&dl=ACM&CFID=1518545&CFTOKEN=40225712>.

Roman, E. 1999. Mastering Enterprise JavaBeans and the Java 2 Platform, Enterprise Edition. New York: John Wiley & Sons, Inc.

Romanovsky, A. 2001. Coordinated Atomic Actions: How to Remain ACID in the Modern World. ACM SIGSOFT Software Engineering Notes, vol. 26, iss. 2, March 2001, 66-68. ACM Press, New York, NY, USA. [Viitattu 9.5.2002]. Saatavana www-osoitteessa <http://delivery.acm.org/10.1145/510000/505793/p66-romanovsky.pdf?key1=505793&key2=3069520201&coll=portal&dl=ACM&CFID=2471911&CFTOKEN=28153729>.

Shim, S. S. Y & Sundaram, M. 2001. Infrastructure for B2B Exchanges with RosettaNet. Third International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems, WECWIS 2001, 110-119. [Viitattu 9.6.2002]. Saatavana www-osoitteessa <http://ieeexplore.ieee.org/iel5/7428/20209/00933912.pdf?isNumber=20209&prod=CNF&arnumber=933912&arSt=110&ared=119&arAuthor=Sundaram%2C+M.%3B+Shim%2C+S.S.Y.>.

Szyperski, C. 1998. Component Software: Beyond Object-Oriented Programming. Addison-Wesley, New York ACM Press.

Szyperski, C. 2001. Components and web services, why are emerging transaction standards so close to what we have seen? Software Development Online. [Viitattu 7.11.2001]. Saatavana www-osoitteessa <http://www.sdmagazine.com/documents/s=844/sdm0108c/0108c.htm>.