Yritysten väliset yhteistoiminnan hallintasopimukset ja hallintapalvelut

Janne Metso

Helsinki 26. toukokuuta 2006
Pro gradu -tutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Matemaattisluonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Janne Metso

Työn nimi — Arbetets titel — Title

Yritysten väliset yhteistoiminnan hallintasopimukset ja hallintapalvelut

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro Gradu	26. toukokuuta 2006	63 + 2

Tiivistelmä — Referat — Abstract

Yritykset käyttävät sähköisiä tietojärjestelmiä entistä enemmän liiketoiminnan prosessien toteuttamiseen ja avustamiseen. Väliohjelmiston näkökulmasta yhteistoimintaa voidaan hallinta teknisill sopimuksilla, jotka vastaavat sovellusten yhteentoimintaan niin kuljetuskerroksella, sovellusten käyttäytymisen tasolla kuin sovellusten semantiikankin tasolla. Kukin sopimus tarjoaa tietovaraston, jonka avulla sopimusosapuolet voivat muokata väliohjelmiston asetukset sellaisiksi, että yhteistoiminta onnistuu halutulla tavalla. Yritysten välistä yhteistoimintaverkostoa hallitseva järjestelmä tarjoaa välineeet sopimusten elinkaaren hallintaan sopimuksen luomisesta ja neuvottelusta sen umpeutumiseen.

Tässä tutkielmassa on kartoitettu sopimusten tietosisältöä ja tietojen käyttö teknisen yhteistoiminnan varmistamiseksi. Erityisesti on toteutettu prototyyppijärjestelmä, joka mahdolistaa automaattisen sopimusten elinkaaren hallinnan.

Computing Reviews Classification:
D.2.12 Interoperability, H.5.3 Group and Organization Interfaces, C.2.4 Distributed Systems

Avainsanat — Nyckelord — Keywords
yhteistoiminta, sähköiset sopimukset, sopimusten hallinta

Säilytyspaikka — Förvaringsställe — Where deposited
Kumpulan tiedekirjasto, sarjanumero C-2006-

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

Liitteet

Luku 1

Johdanto

Internet on mullistanut liiketoimintaa mahdollistamalla yritysten tietojärjestelmien väliset yhteydet ja laajentamalla yritysten liiketoiminta-aluetta. Internetin välityksellä kaksi yritystä voi vaihtaa tietoja keskenään sähköisesti aikaisemmin käytettyjen faksien ja puhelinten sijaan. Internet on avannut mahdollisuuden sähköiseen kaupankäyntiin kauempanakin olevien yritysten kanssa standardoimalla sähköisen kommunikaation alemmat kerrokset.

Yritykset muodostavat keskenään virtuaaliorganisaatioita keskittyäkseen omaan ydinliiketoimintaansa ja tuottaakseen suurempia palvelukokonaisuuksia. Virtuaaliorganisaatiossa pienet, erikoistuneet yritykset tuottavat yhdessä isomman kokonaisuuden hyödyntäen kaikkien virtuaaliorganisaation osapuolten osaamista ja siten hyödyttävät kaikkia osapuolia. Virtuaaliorganisaatioiden avulla voidaan myös laajentaa tarjotun tuotteen tai palvelun markkina-aluetta yhdistämällä useita eri alueilla toimivia paikallisia yrityksiä keskenään.

Yhden tai useamman yrityksen tarjotessa yhteisiä palveluita on otettava huomioon liiketoiminta-alueiden väliset erot ja yritysten väliset semanttiset erot. Eroja muodostavat esimerkiksi kansalliset lait, jotka on keskenään erilaisia. Eri liiketoiminta-alueilla käytetään myös omia vakiintuneita käytäntöjä, jotka eroavat muiden alueiden käytännöistä. Liiketoiminta-alueiden väliset erot on otettava huomioon rakennettaessa virtuaaliorganisaatiota ja sovitettaessa kahta tietojärjestelmää yhteen.

Palveluorientoitunut laskenta (*service oriented computing*, SOC) ja palveluorientoituneet arkkitehtuurit (*service oriented architectures*, SOA) mahdollistavat liiketoiminnan kuvaamisen palveluksi ja palveluiden koostamisen isommaksi kokonaisuudeksi. Palveluorientoituneet arkkitehtuurit tarjoavat lisäksi apupalveluita liiketoimintapalveluiden löytämiseksi ja koostamiseksi.

Virtuaaliorganisaatioon osallistuvien yksittäisten yritysten tietojärjestelmät tulee yhdistää toimivaksi kokonaisuudeksi, jotta virtuaaliorganisaatiosta on mahdollisimman paljon hyötyä osallistujilleen. Yksittäiset osallistujat ovat itsenäisiä toimijoita, jotka tekevät päätöksiä liiketoimintastrategiasta ja käyttävät omia tietojärjestelmiään. Tietojärjestelmien toteutustekniikat ja sovellusalustat (kuten CORBA [42] tai J2EE [21]) vaihtelevat yritysten välillä. Tämän seurauksena järjestelmät toimivat erilaisilla tavoilla ja tarjoavat erilaisia rajapintoja ulkopuolisten käyttöön.

Heterogeenisessa ympäristössä toimivien virtuaaliorganisaatioiden tietoteknistä yhteistoimintaa (*interoperability*) säädellään sopimusten avulla. Sopimus muodostetaan kaikkien organisaation osallistujien kesken neuvotteluprosessin avulla. Vaikka kahdenväliset sopimukset ovatkin suositeltavia [55], monenvälisiäkin sopimuksia voidaan käyttää tilanteessa, jossa useampi yritys tarjoaa palveluita yhdessä. Yhteinen toiminta asettaa kaikille osapuolille yhteisiä vaatimuksia, jotka kootaan sopimukseen. Yhteisellä sopimuksella varmistetaan vaatimusten todellinen yhtenäisyys kaikkien osapuolten kesken. Käytettäessä kahdenvälisiä sopimuksia yhtenäisyydestä ei voida olla varmoja.

Sopimukset sisältävät informaatiota yhteistoiminnan varmistamiseksi eri osallistujien kesken ja tietosisältöön kuuluu palveluiden teknisiä ja semanttisia kuvauksia. Sopimukselta ja siihen sisältyviltä erilaisilta yhteistoimintaan liittyviltä kuvauksilta edellytetään validoitavuutta ja verifioitavuutta. Validointi lisää osallistujien luottamusta, ja etukäteen verifioidut tekniset kuvaukset helpottavat järjestelmän seurantaa. Verifiointi on sitä tärkeämpää mitä suurempi ja monimutkaisempi virtuaaliorganisaation rakenne on. Koska internetissä toimittaessa virtuaaliyrityksen osapuolet voivat olla hyvinkin epäluotettavia ja koska verifioinnilla ei kuitenkaan pystytä varmistamaan tietojärjestelmien sataprosenttista yhteistoimintaa, pitää sopimuksen avulla voida myös valvoa sekä koko virtuaaliyrityksen toimintaa että yksittäisten yritysten toimintaa virtuaaliyrityksen sisällä.

Tekniset yhteistoimintasopimukset tarvitsevat ympärilleen tukipalveluja. Tukipalveluja tarjotaan erillisen hallintaympäristön avulla. Sopimuksen elinkaareen kuuluu sopimuksen luominen, siitä neuvotteleminen ja sen käyttäminen. Hallintaympäristön tulee tarjota tuki kaikille näille elinkaaren vaiheille. Hallintaympäristön tulee myös tukea sopimuksen toimeenpanoa.

Tutkielman tavoitteina on määritellä yhteistoimintasopimukselle vaadittava tietosisältö, sopimuksen hallintaympäristön vaatimukset, verrata kahta olemassa olevaa ratkaisua vaadittavaan tietosisältöön ja suunnitella ja toteuttaa prototyyppi sopimusten hallintaympäristöstä sekä toteuttaa tietorakenne sopimuksen tallentamiseksi. Tietosisältö määritellään lähteisiin perustuvan tarvekartoituksen avulla. Tarvekartoituksen keskeisiä tavoitteita on saavuttaa riittävä tietomäärä yhteistoiminnan varmistamiseksi ja mahdollistaa tiedon verifioitavuus, valvottavuus ja monikäyttöisyys. Verifioitavuudella tarkoitetaan sopimukseen sisältyvien yhteistoimintakuvausten verifioimista yhdessä toimiviksi. Valvottavuudella tarkoitetaan sitä, että yritykset voisivat sisäisten järjestelmiensä avulla valvoa omasta puolestaan sekä omaa että muiden käyttäytymistä ja todeta onko se sopimuksen mukaista vai ei. Sopimuksen monikäyttöisyydellä tarkoitetaan tämän tutkielman puitteissa saman sopimusmallin käyttämistä usealla eri liiketoiminta-alueella. Käytettäessä samaa sopimusmallia eri liiketoiminta-alueilla tai jopa eri valtioissa tarvitaan standardoituja sopimusosia, joita voidaan valita tarpeen mukaan. Sopimusosat valitaan liiketoiminta-alueen tai valtion lakien perusteella. Tarvekartoituksen tuloksena syntynyt sopimuksen tietosisältö pyritään muokkaamaan XML-pohjaiseen[1] muotoon XML-skeeman[2] avulla. Sa-

[1]XML standardi löytyy osoitteesta http://www.w3.org/XML

[2]XML Skeeman standardi löytyy osoitteesta http://www.w3.org/XML/Schema

malla myöskin tuetaan omalla tavallaan sopimuksen monikäyttöisyyden tavoitetta, koska XML on toteutuksesta riippumaton tapa esittää dokumentteja.

Olemassa olevina järjestelminä ja standardeina käytetään ebXML- ja OWL-S -standardeja. Standardeja verrataan kirjallisuuden määrittämiin sopimuksen sisällön tarpeisiin ja sen tavoitteena on tarkistaa olemassa olevien järjestelmien riittävyys todelliseksi sopimuspohjaksi. Olemassa olevia järjestelmiä käytetään myös esimerkkeinä oman ratkaisun suunnittelussa ja toteutuksessa.

Tutkielmaan liittyy suunniteltava ja toteutettava ohjelmiston osa, joka sisältää sopimuksien hallintaan tarkoitetun ympäristön ja tietorakenteen johon sopimuksen tietosisältö tallennetaan. Hallintaympäristö koostuu kahdesta komponentista verkostonhallinta-agentista ja sopimusvarastosta. Suunnittelussa on käytetty lähtökohtana web-Pilarcos -projektissa [3, 27, 28] suunniteltua ja toteutettua prototyyppiympäristöä. Edellä mainitut komponentit ja tietorakenne on toteutettu osaksi mainittua prototyyppiä.

Tämän tutkielman rakenne on seuraava. Luvussa 2 määritellään yhteistoiminnan hallintasopimuksille yleistä viitekehystä ja pureudutaan sopimukselle, sen sisällölle ja sopimuksia tukevalle tekniselle ympäristölle oleviin vaatimuksiin. Vaatimusmäärittely on kirjallisuuslähtöistä. Luvussa 3 määritellään vaatimuksien pohjalta hallintasopimuksille tietosisältö. Tietosisältöön kuuluu sekä teknisiä aspekteja hallintaympäristöä varten että yhteistoiminnan varmistamiseen pyrkiviä osakokonaisuuksia. Luvussa 4 esitellään web-Pilarcos -projektissa tehdyn prototyypin osaksi tuotettu sopimusten hallintaympäristö, joka tukeutuu esiteltyyn sopimuksen tietosisältöön ja aiemmin esiteltyihin teknistä ympäristöä koskeviin vaatimuksiin. Teknisen ympäristön jälkeen luvussa 5 pohditaan asetettujen vaatimusten toteutumista, sopimusten hallintaan liittyviä ongelmia sekä mietitään järjestelmän suorituskykyä.

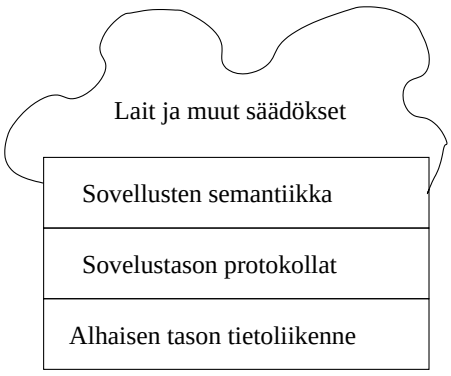
Luku 2

Tausta ja vaatimukset

Sopimukset ovat olleet perinteisesti selväkielisiä asiakirjoja, jotka on valmisteltu pitkän prosessin tuloksena. Sopimuksille merkittävä ominaisuus on niiden laillinen sitovuus ja sovitun sisällön varmistaminen. Sopimuksiin vaikuttaa voimakkaasti se lakiin pohjaava ympäristö, jossa ne on solmittu. Esimerkiksi Suomessa laki ei juurikaan rajoita sopimuksen muotoa kahden yrityksen välillä. Toisaalta yrityksen ja yksityisen henkilön välisiä sopimuksia säädellään kuitenkin hyvin tarkasti. Tilanne vaihtelee maasta toiseen voimakkaasti.

Teknisten yhteistoimintasopimusten tehtävä on sisältää informaatiota liittyen tietojärjestelmien yhteistoimintaan ja niiden yhteistoiminnan varmistamiseen. Samalla ei kuitenkaan haluta luopua olemassa olevista ja suuria investointeja vaatineista järjestelmistä. Tietojärjestelmät pitää kuitenkin saada toimimaan yhteen liiketoiminnan mahdollistamiseksi.

Tietojärjestelmien yhteistoiminnan tutkimus on keskittynyt teknisten osien ympärille. Tämän hetkinen tutkimuksen tila on esitetty kuvassa 2.1. Kuvassa on neljä tasoa: alhaisen tason tietoliikenne, sovellustason protokollat, palveluiden semantiikka ja ylimpänä lait ja muut säädökset. Alhaisen tason tietoliikenteellä tarkoitetaan muun muassa kuljetuskerrosta ja vastaavia tekniikoita. Sovellustason protokollat tar-



Kuva 2.1: Yhteistoimintaan liittyvän tutkimuksen tila.

koittavat esimerkiksi työnkulunkuvauskielillä mallinnettua sovelluksen ulkoista käyttäytymistä. Palveluiden semantiikalla tarkoitetaan kuvausta palvelun tehtävästä ja sen tuottaman informaation luonteesta. Tällä hetkellä alhaisen tason tietoliikenteen ongelmat on aika pitkälle ratkaistu ja ymmärystä on aika paljon sovellustason protokollista sekä palveluiden semantiikan kuvaamisesta. Lakien ja muiden säädösten huomioon ottaminen on kuitenkin kesken ja vasta kehittymässä [1].

Tässä osassa pohditaan aluksi sopimuksiin liittyviä tavoitteita sekä esitellään olemassa olevia ratkaisuita sähköisiin sopimuksiin. Esimerkkitilanne esitellään tavoitteiden ja tietosisällön hahmottamisen helpottamikseksi. Esimerkin jälkeen pohditaan lain asettamia vaatimuksia sopimuksille ja seuraavaksi kootaan sopimusten tietosisältöön liittyviä vaatimuksia. Sopimuksen käsittelyn kautta päästään elinkaareen ja sähköisiä sopimuksia tukevan ympäristön vaatimuksiin.

2.1 Sähköisten sopimusten luonteesta

Sopimuksen avulla muodostetaan suhde kahden tai useamman osapuolen välille siten, että muodostuu yhteinen hyväksyntä [33]. Yhteinen hyväksyntä muodostuu sopimukseen kirjatuista seikoista, jotka sähköisten sopimusten tapauksessa liittyvät tehtävien (*task*) [33] tai sähköisten palveluiden käyttämiseen tietyssä virtuaaliorganisaatiossa. Virtuaaliorganisaatiolla tarkoitetaan joukkoa itsenäisiä organisaatioita, jotka toimivat yhdessä jonkin tietyn päämäärän saavuttamiseksi.

Osapuolten määrä sopimuksessa vaikuttaa sen monimutkaisuuteen. Myös osapuolten väliset suhteet muodostuvat monimutkaisemmiksi. Yksinkertaisuuden vuoksi suositellaankin kahdenvälisiä sopimuksia [55]. Toisaalta kahdenväliset sopimukset rajoittavat huomattavasti mahdollisten virtuaaliorganisaatioiden kokoa. Jos useamman kuin kahden väliset sopimukset puretaan useaksi kahdenväliseksi sopimukseksi, menetetään yhteinen vastuu päämäärän saavuttamisesta. Samalla menetetään myös mahdollisuus mallintaa usean osapuolen toiminnasta riippuvia transaktioita.

Yksi merkittävimpiä haasteita sopimusten mallintamisessa sähköisesti on se, että todellisen maailman (*real world*) luonnolliset käsitteet eivät ole normaalisti tulkittavissa automaattisesti [54]. Tämän kaltaisia käsitteitä ovat esimerkiksi hevonen ja auto. Yleisesti myös sopimuksissa käytetään vakiintuneita laki-ilmaisuja kuvaamaan asioita tai toimijoita kuten asiakasta. Haaste muodostuu erityisesti sellaisissa sopimuksissa, joissa käsitellään myös materiaalivirtoja pelkkien sähköisten palveluiden tai tehtävien lisäksi. Yksi ratkaisu on muodostaa synonyymisanasto avustamaan tietokoneen päättelyä sopimustekstistä.

Sähköisen liiketoiminan ratkaisu EDI (Electronic Data Interchange) on ollut saatavilla jo pitkään, mutta sen ongelmana on kalleus ja kankeus [54]. Tämä tarkoittaa sitä, että sen käyttäminen nykyisessä taloudessa on hankalaa nopeasti vaihtuvien yhteistyökumppaneiden kanssa. Suomen mittakaavassa pienille ja keskisuurille yrityksille EDI on aivan liian kallis.

Viime vuosina on tehty useita aloitteita yritystenvälisen yhteistoiminnan hallintaan ja sopimusten käyttämiseen osana tätä hallintaa [9]. Aloitteisiin kuuluu muun

muassa ebXML [36] ja sen tekninen arkkitehtuuri [11], OBI (Open Buying on the Internet) [40, 49], OWL-S (Ontology Web Language - Services) [31] ja BCA (Business Contracts Architecture) [34, 9]. Näillä aloitteilla on pyritty helpottamaan ja tutkimaan yritysten välistä yhteistoimintaa tietokoneavusteisesti. Aloitteet ovat pyrkineet määrittelemään yhteistoiminnan saavuttamiseen tarvittavat tiedot ja sopimuksen muodon. Aloitteet lähestyvät ongelmaa kuitenkin kovin eri näkökulmista ja niiden välillä ei oikeastaan ole havaittavissa konvergenssia.

ebXML on alun perin YK:n ja sittemmin OASIS:n[1] standardointiprojekti, jolla pyritään saamaan helppo ja halpa sähköisen liiketoiminnan ratkaisu. Lähtökohtana on ollut avoimempi ja halvempi ratkaisu kuin EDI, jotta sitä voidaan käyttää myös kehitysmaissa. Tärkeimmät rakenteet ebXML:ssä on kuvattu XML-kielellä. Näitä ovat muun muassa viestintä, liiketoimintaprosessit, palveluhakemisto ja erilaiset yritysprofiilit. Yritysprofiileita käytetään sopimusten muodostamisessa ja niiden sisältöä käsitellään tarkemmin luvussa 2.3. Koko liiketoimintaan tarvittava teknologiapino kuljetuskerroksesta palveluiden kuvaamiseen ja palveluiden kokoamiseen määritellään ebXML:llä yhdeksi toimivaksi kokonaisuudeksi. ebXML:n metamalli tarjoaa yksinkertainen kehyksen taloudellisen perustan omaavien sopimusten mallintamiseen [9].

OBI [40, 49] on tuntemattomampi aloite sähköiseen liiketoiminnan toteuttamiseen. Sen tavoitteena on olla toimittaja- ja toteutusalustaneutraali lähestymistapa alhaisen tason sähköisten palveluiden yhteistoimintaan. Tavoitteena on myös luoda maailmanlaajuinen metodologia sähköisen kaupan helpottamiseksi. Se määrittelee tiedon muotoon, kuljetukseen ja turvallisuuteen liittyviä osia yhteistoiminnasta.

OWL-S [31] on alun perin Semantic Web[2]-aloitteen pohjalta kehitetty ontologiakieli. Sen pohjalla on Yhdysvaltojen puolustusministeriön DARPA-osaston DAML-kieli (DARPA Agent Markup Language). OWL-S:n tarkoituksena on tarjota kieli, jonka avulla palveluita voidaan kuvata yksiselitteisesti ja tietokoneen ymmärtämässä muodossa. Kuvauksiin kuuluvat palveluiden ominaisuudet ja kykeneväisyydet (*capabilities*). Kuvausten ideana on tarjota mahdollisuus esittää palveluiden ominaisuudet niin tarkasti, että palveluita voidaan automaattisesti valita käytettäväksi OWL-S-kuvausten perusteella. Kielen avulla voidaan muodostaa palveluista ja näiden ominaisuuksista ontologiaverkostoja. Näiden verkostojen avulla voidaan päätellä palvelun sopivuus tiettyyn käyttötarkoitukseen. Edellä mainituista esimerkeistä ebXML:ää ja OWL-S:ää tullaan käyttämään vertailukohtina sopimuksen sisällön vaatimusten käsittelyn yhteydessä luvussa 2.3.

BCA on Australian DSTC-instituutissa[3] kehitetty tutkimusprototyyppi, joka keskittyy sähköisten sopimusten mallinnukseen ja siihen liittyviin ongelmiin. Sopimusten mallinnus perustuu deonttisen logiikan tapaisiin käsitteisiin. BCA:n tavoitteena on mallintaa selkokielisistä sopimuksista tietokoneen ymmärrettäviä sähköisiä sopimuksia, joita voidaan valvoa automaattisesti. BCA:n arkkitehtuuri koostuu pääosin neljästä komponentista [45]. Nämä ovat Contract Repository (CR), Notary,

[1] http://www.oasis-open.org/

[2] http://www.semanticweb.org

[3] http://www.dstc.edu.au

Contract Monitor (CM) ja Contract Enforcer (CE). CR tallentaa sopimusten osia käytettäväksi sopimusten muodostamisen yhteydessä, Notary toimii valmiiden sopimusten säilystyspaikkana ja on kolmannen, luotetun osapuolen toteuttama, CM valvoo kahden palvelun välistä tietoliikennettä ja CE:n tehtävä on sopimuksen toimeenpano CM:n tuottaman tiedon perusteella.

Merkittävinä edellä mainittuja hankkeita edistävinä seikkoina nähdään muun muassa XML:n käyttöönottaminen, Internet ja hajautettuihin olioihin perustuvat teknologiat [9]. XML:n käyttö helpottaa yhteisenä alustana muun muassa kommunikoinnissa (web-palvelut) ja tarjoaa yleisesti käytössä olevan koodauksen informaation ilmaisemiseen. Pelkkä XML ei kuitenkaan takaa sitä, että kaksi osapuolta ymmärtävät siirretyn tiedon samalla tavalla. Hajautettuihin olioihin perustuvat teknologiat eivät myöskään itsessään helpota yhteistoimintaa, vaan tarjoavat välineitä oikeanlaisten sovellusten tekemiseen. Hajautus tarvitsee aina yhteisen kommunikointikielen toimiakseen. XML ja hajautetut olioteknologiat kuitenkin muodostavat yhdessä hyvän kokonaisuuden. Internet on puolestaan mahdollistanut uudenlaisen kommunikoinnin kaukanakin toisistaan olevien yritysten kesken poistaen siten toimipaikkaan liittyviä rajoituksia.

Tiedon yhtenäisyys on merkittävä ongelma hajautetuissa ja heterogeenisissa olosuhteissa [51]. Ristiriidat tiedon yhtenäisyydessä voidaan jakaa kahteen luokkaan: attribuuttien arvojen ristiriitoihin ja konfigurointi ristiriitoihin. Ohjausjärjestelmien attribuuttien arvojen ristiriitoja syntyy, kun saman attribuutin arvo tallennetaan useisiin rinnakkaisiin järjestelmiin. Kun sitten yhden järjestelmän paikallisen käyttöliittymän avulla muutetaan attribuutin arvoa, se ei välity muihin järjestelmiin. Konfigurointi ristiriidat liittyvät järjestelmien kokoonpanoihin ja niiden muutoksiin. Nämä muutokset eivät välttämättä päivity kaikkiin järjestelmiin kerralla.

Ajateltaessa sopimusten mallinnusta ja yhteistoimintaa, on muistettava, että osallistujien asymmetrisyys on fundamentaalia [55]. Kahden osapuolen välillä aina toinen kontrolloi tilannetta. Yksi esimerkki tästä on palvelin-asiakas suhde. Jopa tasavertaisten osapuolten välillä aina toinen suorittaa jotain palvelua tai on vastuussa kontrollin siirtämisestä vastapuolelle.

Sopimuksia luotaessa luomiseen liittyviä palveluita käytetään aina tietyn sopimustoimialueen sisällä [47]. Sopimustoimialueella tarkoitetaan tässä tilanteessa sitä aluetta, jonka lait ja muut säädökset rajoittavat sopimusta. Esimerkiksi Suomi lakeineen on tällainen sopimustoimialue. Tämä toimialue muodostaa sopimukselle ja sen luomisessa käytetyille palveluille tietynlaisen kontekstin, jonka puitteissa muodostus tapahtuu. Toimialue vaikuttaa myös tapoihin, joilla sopimuksen toimeenpano varmistetaan ja valvotaan sitä, että osapuolet noudattavat sopimusta. Toimialue myös osaltaan määrittelee erilaiset mahdollisuudet, joilla erimielisyyksiä sopijaosapuolten kesken voidaan ratkoa.

Ensimmäinen näkökulma järjestelmien turvallisuuteen on se, että yritysten sisäiset politiikat ovat usein luottamuksellisia. Tämä mutkistaa yhteistoimintaa, sopimuksia ja valvontaa, koska tällöin politiikoita ei haluta antaa muiden, edes sopimuskumppaneiden, käsiteltäväksi [33]. Tämä monimutkaistaa asioita, koska järjestelmän ja sopimuksen pitää pystyä suojaamaan osapuolten sisäisiä politiikoita. Tämä vaa-

tii sopimuksen osalta osittaista salaamista siten, että tietyt osat ovat nähtävissä vain rajoitetulla joukolla osapuolia. Toisaalta yrityksen käyttäytymisestä voidaan päätellä sen sisäisiä politiikkoja ja mieltymyksiä.

Toinen näkökulma järjestelmien turvallisuuteen on se, että usein eri toteutusalustoilla on käytössä epäyhtenäisiä turvallisuuspolitiikkoja [6]. Avoimien hajautettujen järjestelmien tapauksessa pitäisi olla mahdollista päätellä koko järjestelmästä sen turvallisuuteen liittyviä yksityiskohtia. Kun järjestelmä koostuu potentiaalisesti useista autonomisista yksiköistä, pitäisi pystyä varmistamaan koko järjestelmän turvallisuus. Sopimuksessa määriteltyjen turvallisuusvaatimusten avulla ilmaistaan yhtenäinen turvallisuustaso koko virtuaaliorganisaatiolle.

2.2 Esimerkki

Esimerkkitilanteen tarkoitus on auttaa lukijaa hahmottamaan tässä tutkielmassa käsiteltävien sopimusten tavoitteita ja tietosisällön tarpeita. Tavoitteena on myös konkretisoida, millaisiin tilanteisiin tässä tutkielmassa esiteltäviä sopimuksia käytetään. Myöhemmin esiteltävä tietosisältö sidotaan esimerkkitilanteeseen ja esitetään, mitä kullakin tiedolla esimerkin valossa saavutetaan.

Taulutelevisioita valmistava Suomen Telkku OY tarvitsee televisioiden valmistamiseen näyttöpaneelin ohjauskortteja. Muut osat, kuten näyttöpaneelit, yritys valmistaa itse. Toimintojensa tehostamiseksi Suomen Telkku OY on ottanut käyttöön sähköisen liiketoimintaprosessien ohjausjärjestelmän. Muun muassa komponenttien ostaminen on mallinnettu uuden järjestelmän avulla ja yhdistetty suoraan yrityksen tehtaan varastokirjanpitoon. Ohjausjärjestelmän ulkoiset liittymät on toteutettu web-palvelu-tekniikalla.

Löytääkseen sopivan ohjainkorttien valmistajan Suomen Telkku OY:ssä on tehty mittava selvitys mahdollisista kumppaneista. Ensisijalla valintakriteereissä oli kotimainen toimittaja, mutta sellaista ei ollut saatavilla. Kaksi eri vaihtoehtoa kuitenkin löytyi. Ensimmäinen niistä on Ruotsissa toimiva Svensk Elektronik AB ja toinen Brasiliassa toimiva Hugos Henchmen Inc. Valinta ei ole helppo, sillä Ruotsista tulevat ohjainkortit ovat laadukkaampia, mutta kalliimpia, ja Brasiliasta tulevat kortit huomattavasti edullisempia. Pontentiaalisilta kumppaneilta edellytetään myös sähköisten liiketoimintaprosessien ohjausjärjestelmää.

Suomen Telkku OY haluaa muodostaa virtuaaliorganisaation uuden ohjainkorttien toimittajan kanssa. Koska molemmat ehdokkaat ovat kaukana Suomen Telkku OY:n tehtaista, kuljetus joudutaan jotenkin järjestämään, mutta ohjainkorttien valmistajan luotetaan hoitavan kuljetukset Suomeen. Koska tilauksia tehdään isommissa erissä, käytetään merikuljetuksia. Tämäkään ei siis tuo selkeää ratkaisua ohjainkorttien toimittajan valintaan.

Koska ehdokkaat ovat paperilla samantasoisia, Suomen Telkku OY päättää neuvotella molempien kanssa toimitusten ehdoista. Neuvotteluiden aikana otetaan esille käytössä oleva prosessien ohjausjärjestelmä ja siihen mallinnetut erilaiset ostoprosessit. Neuvotteluissa käy ilmi, että molemmat ehdokkaat pystyvät vastaanottamaan

tilauksia sähköisessä muodossa. Ruotsalaisyritys luottaa J2EE-tekniikkaan ja Brasiliassa on panostettu web-palvelu-tekniikkaan. Ruotsalaiset kuitenkin lupaavat tukea tarvittaessa myös web-palveluita.

Neuvotteluiden edetessä käy selväksi, ettei Brasiliassa sijaitsevalla Hugos Henchmen Inc:llä ole tarvittavaa kokemusta halutunlaisten ohjauskorttien valmistamiseen. Myös sähköinen prosessinohjausjärjestelmä ja sen tarjoamat rajapinnat ovat täysin epäyhteensopivia Suomen Telkku OY:n omien teknisten vaatimusten kanssa. Näin Svensk Elektronik AB valitaan ohjauskorttien toimittajaksi.

Koska Ruotsalaisen yrityksen tietojärjestelmät on toteutettu J2EE-tekniikalla, yritys tilaa alihankkijalta web-palvelu-kääreet (*wrapper*) omaan järjestelmäänsä. Kääreiden avulla yritys pystyy tarjoamaan web-palvelurajapinnan omiin järjestelmiinsä. Kääreiden toiminta määritellään tiukasti noudattamaan Suomen Telkku OY:n ostoprosessia, jotta teknisiä ongelmia on mahdollisimman vähän.

Sopimuksen muodostuksen jälkeen yritykset alkavat toimia yhdessä. Ensimmäisen tilauksen yhteydessä paljastuu ensimmäinen ongelma tietojärjestelmien yhtenäisyydessä. Svensk Elektronik AB:n lähettämässä laskutus-viestin tyyppi ei täsmää Suomen Telkku OY:n järjestelmien käyttämän tyypin kanssa. Kenttiä on erilainen määrä, eivätkä niiden nimetkään täsmää keskenään. Kääreiden toiminassa on siis kuitenkin virheitä ja jotain jäi määrittelemättä. Ongelma korjataan ja ensimmäinen toimitus saadaan lähtemään.

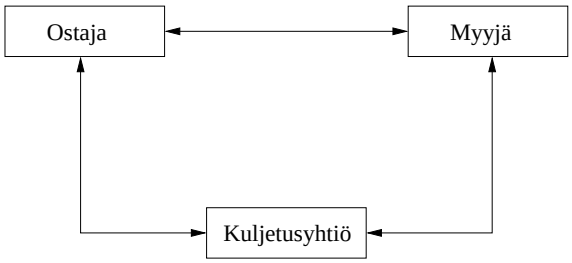
Toimitus lähetetään ruotsalaisen yrityksen valitseman kuljetuspalvelun kautta. Seuraavaan ongelmaan törmätään, kun toimitus saapuu perille ja sitä ei ole kukaan vastaanottamassa. Vastaanottaja puuttuu, koska toimitus saapuu viikonlopun aikana ja kuljetusyrityksen lähettämää ilmoitusta toimituksen saapumistesta ei kyetty vastaanottamaan vääräntyyppisen viestin vuoksi.

Seuraavat toimitukset saapuvat jo perille ongelmitta. Koska Svensk Elektronik AB:n ensimmäinen valinta kuljetusyritykseksi ei ollut sopiva, yritys joutui valitsemaan uuden kuljetusyhtiön tarkempien teknisten vaatimusten perusteella, kun alunperin oletettiin. Suomen Telkku OY:n ja kuljetusyrityksen tietojärjestelmien yhteensopivus olisi pitäny tarkistaa ennen valintaa.

Kahden vuoden toiminnan jälkeen ohjainkortit muuttuvat olennaisesti ja yritykset päättävät luoda uuden sopimuksen vanhan pohjalta. Uuden sopimuksen kustannukset määritellään uudelleen ohjainkortin valmistuskustannusten mukaan. Samalla vanha sopimus puretaan.

Esimerkkitilanteen tekninen arkkitehtuuri voidaan esittää kolmen roolin avulla. Nämä ovat ostaja, myyjä ja kuljetusyhtiö. Roolit ja niiden väliset suhteet on esitetty kuvassa 2.2. Kaikki roolit liittyvät toisiinsa, koska tietojärjestelmät siirtävät tietoa toisiinsa. Ostaja ja myyjä vaihtavat tilaustietoa, myyjä ja kuljetusyhtiö sopivat kuljetuksen yksityiskohdista ja ostaja vastaanottaa tietoa kuljetusyhtiöltä kuljetuksen eri vaiheista.

Virtuaaliorganisaation osallistujamäärä vaihtelee vaiheesta toiseen. Aluksi organisaatiossa ei ole kuin ostaja ja myyjä. Kuljetusyhtiö otetaan mukaan vasta ensimmäisen kuljetuksen yhteydessä. Kuljetusyhtiö voidaan myös päättää kuljetuskoh-



Kuva 2.2: Esimerkkitilanteen tekninen arkkitehtuuri [15].

taisesti. Tärkeää on kuitenkin se, että kuljetusyhtiö pystyy liittymään ostajan ja myyjän tietojärjestelmiin.

Teknisiltä yksityiskohdiltaan edellä esitellyn virtuaaliorganisaation kaltainen tilanne on mallinnettu Web-Pilarcos -projektissa [15]. Projektin tilanne mallinnettiin BPEL-kuvauksilla ja Java^{TM}-olioilla. Nämä kuvaukset vastaavat edellä esitellyn esimerkin tilannetta, mutta eri toimijoiden kuvauksien välillä ei ole yhteentoimivuusongelmia, koska kaikki toimijat on mallinnettu yhtäaikaa. Eri osapuolten tietojärjestelmät on toteutettu täysin erillisiksi ja ne käyttävät omia toimtakuvauksiaan palveluidensa luomiseksi. Muita esimerkkitilanteen teknisiä yksityiskohtia, kuten ulkoisia toimintakuvauksia, käsitellään erikseen tietosisällön yhteydessä luvussa 3.

Esimerkin tapauksessa sopimukseen neuvoteltiin virtuaalorganisaation rakenne roolien avulla, jokaisen osapuolen sähköisestä liiketoimintaprosessista ulkoiset osat ja yritysten välillä lähetettävät viestien tyypit. Prosessin ulkoiset osat sisältävät toiselle yritykselle tarjottavat operaatiot ja operaatioiden järjestyksen. Prosessien ja viestien lisäksi sopimuksessa määritellään yhteinen alhaisen tason kommunikointiprotokolla. Tukipalveluina esimerkissä tarvitaan osapuolten hakupalveluita, tietokoneavusteista neuvottelua, sopimuksen validointia ja voimassaolon aikaista valvontaa.

2.3 Sopimuksen sisältö

Mallinnettaessa sähköisiä yhteistyönhallintapalveluita sopimusta pidetään avain elementtinä kaikkien laista riippuvien elementtien tallentamiseksi [47]. Kaupallisten transaktioiden osalta sopimuksessa voidaan mallintaa erilaiset roolit ja käyttää transaktioita näiden välisen informaatiovuon mallintamiseen.

Sopimuksia tulisi käyttää e-kaupassa osapuolten riskitason minimoimiseksi aivan kuten normaalissakin kaupankäynnissä [30]. Nousevat yritysten väliseen kaupankäynnin teknologiat tekevät sopimusten tukemisen mahdolliseksi. Sopimusten tukemiseksi tarvitaan joukko erilaisia palveluita kuten validointia, sopimusten osapuolten etsimistä ja neuvottelua.

Osapuolten riskin minimoimiseksi sopimusten pitää olla laillisesti sitovia [9]. Laillinen perusta vähentää osapuolten riskiä, koska jonkin mennessä vikaan voidaan korvauksia hakea oikeusteitse. Uhka oikeuskäsittelystä lisää keskinäistä painetta sopi-

muksen pykälien noudattamiseen. Laillisessa sopimuksessa myös sovitaan virallinen tapa sopimuksen purkamiseen ja siihen liittyvät yksityiskohdat. Ollakseen laillisesti validi sopimuksen pitää olla yksiselitteinen ja osallistujien pitää olla oikeustoimikelpoisia ja toimivaltaisia.

Sopimukset voidaan kirjoittaa alusta alkaen, mutta kehyksiä (*template*) ja standardoituja osia pitää pystyä tukemaan ja käyttämään hyväksi [9, 17]. Tällä estetään tarve luoda uusi sopimuspohja kaikille erilaisille tapauksille ja saadaan sopimuksesta modulaarisempi. Modulaarisuudesta on hyötyä laillisuuden varmistamisessa, kun standardoidut osat pystytään ennalta validoimaan lakeja noudattaviksi. Sopimusten päivittäminen joko liiketoiminta-alueen kehityksen tai toiminnan yhteydess on myös helpompaa. Modulaarisuutta voidaan myös käyttää hyväksi sopimusten tuottamisessa erilaisille liiketoiminta-alueille. Liiketoiminta-alueilla on mahdollisesti omat erikoislisänsä sopimuksiin ja tietyt vakiintuneet käytännöt. Näitä on mahdotonta ottaa huomioon kaikkien alueiden osalta yksinkertaisessa yleisessä sopimuspohjassa. Jotta erilaisia kehyksiä voidaan käyttää, vaaditaan sopimuksen muodostamiseen käytettäviltä palveluilta varastoja, joihin kehyksiä voidaan tallentaa. Samoin kehyksiä ja muita standadoituja osia pitää pystyä hakemaan edellä mainituista varastoista. Varaston tulisi olla UDDI:n [38] tyylinen, mutta tallentaa sopimusten osia.

Laista riippuva tieto tulisi lisätä sopimuksiin erillisinä moduuleina [47]. Tämä lisää sähköisessä muodossa tallennettujen sopimusten luettavuutta. Laeista riippuvien osien tulkintaan tarvitaan myös hyvin todennäköisesti erillisiä palveluita kuten lakimiehiä tai pitkälle erikoistettuja teknisiä palveluita. Erillisten moduulien käyttö myös helpottaa sopimusten käyttämistä useilla eri alueilla, joissa lait ovat toisistaan poikkeavia. Samaa perusrakennetta voidaan käyttää, mutta vain paikallisista laeista riippuva tieto sisällytetään erillisenä moduulina.

OASIS-standardointiorganisaatiossa kehitetään uutta LegalXML-standardia [37]. Tämän standardin avulla pitäisi voida tulevaisuudessa mallintaa niin sopimukset kuin muutkin lakiin liittyvät seikat yhtenäisellä XML pohjaisella kielellä. Standardin valmistumisen jälkeen tässä tutkielmassa määriteltyä sopimuksen tietosisältöä ja teknistä kuvausta voidaan päivittää ottamaan paremmin huomioon lain asettamat määräykset.

Esimerkkitapauksen osapuolten pitää ottaa huomioon eri liike-toiminta-alueiden lait. Yksi yritys on Suomesta ja toinen Ruotsista. Molemmat maat kuuluvat Euroopan Unioniin. Tässä tilanteessa molempien maiden lait ovat suhteellisen yhtenäisiä johtuen EU:sta. Sekä Suomen Telkku OY:n että Svenskt Elektronik AB:n tai niiden edustajien on oltava oikeustoimikelpoisia, jotta sopimus voidaan ylipäätään sopia. Kun sopimus luodaan alusta alkaen yleisemmäksi, sitä voidaan uudelleenkäyttää osapuolten kesken tulevien sopimusten pohjana.

Sopimuspohjaa luonnosteltaessa sen vähittäisvaatimukset ovat roolit, velvollisuudet, sopimuksen liiketoiminta-alue, palveluiden käyttäytyminen, siirrettävät materiaalit ja erilaiset aikarajat [33, 34]. Rooleilla muodostetaan sopimusten osapuolista virtuaaliorganisaation rakenne, ja roolit yhdistetään toisiinsa kommunikaatiokanavin ja riippuvuuksin. Velvollisuudet määrittelevät roolien vastuut esimerkiksi tiedon säilyttämisessä tai tietyn palvelun tarjoamisessa. Liiketoiminta-alue määrittelee so-

pimuksen käyttöalueen ja tarvittavat erikoispykälät. Jos sopimuksen puitteissa siirretään materiaaleja paikasta toiseen, näiden liikkuminen ja ominaisuudet tulee kirjata sopimukseen. Sopimuksen erilaiset aikarajat käsittävät sopimuksen voimassaolon ja toimitusaikarajat. Aikarajat voidaan kirjata joko absoluuttisesti tai suhteellisesti. Absoluuttiset aikarajat ovat yksikäsitteisiä ja suhteelliset riippuvat jostain toisesta tapahtumasta kuten aloituspäivämäärästä.

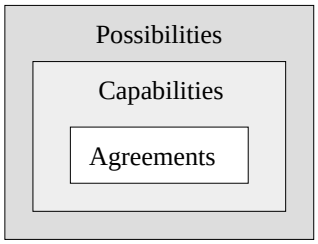
Rooliperustaista virtuaaliorganisaation rakennetta voidaan käyttää myös pääsynvalvontaan ja vastuuvelvollisuuden määrittämiseen [34]. Roolipohjaisella pääsynvalvonnalla (*role based access control*, RBAC) voidaan antaa lupa tiettyihin palveluihin sillä perusteella, että jokin yritys tuottaa tietyn roolin palvelut tietyssä virtuaaliorganisaatiossa. Esimerkiksi auditoijan tulee päästä käsiksi joukkoon resursseja, joita muut osapuolet virtuaaliorganisaatiossa eivät tarvitse. Rooleja voidaan käyttää vastuuvelvollisuuksien määrittämiseen sitomalla tuotetut palvelut rooleihin. Palveluiden sitomisen lisäksi roolin määrityksiin voidaan lisätä erillisiä vastuuvelvollisuuksia. Esimerkiksi tietty rooli voidaan sitoa tallentamaan tietyt asiakirjat tai vastaamaan yleisistä hallintotehtävistä.

Määrittelemällä selkeät roolit vastuualueineen ja käyttäytymiskuvauksineen esimerkin sopimustilanteessa saadaan määriteltyä yksikäsitteisesti tietyn osapuolen tehtävät. Esimerkiksi myyjän tehtävänä on ottaa tilaus vastaan, tuottaa tilauksen mukainen tuote ja varmistaa tilauksen kuljettaminen ostajalle. Toimintakuvauksilla, kuvauksiin liittyvillä viesteillä ja viestien tyypeillä olisi vältetty ensimmäisen tilauksen yhteydessä ilmenneet ongelmat. Määrittelemällä toimitusaikarajoja olisi voitu välttää toimituksen saapuminen sellaiseen aikaan, ettei kukaan ollut paikalla vastaanottamassa.

Sopimukseen voidaan lisätä yhteinen ulkoinen työnkulunkuvaus helpottamaan ymmärrystä sopimuksenalaisesta toiminnasta [55]. Ulkoisella työnkululla tarkoitetaan kaikkien osapuolten palveluiden ulkoista käyttäytymistä integroituna yhdeksi kuvaukseksi. Ulkoiseen käyttäytymiseen kuuluu vain organisaatioiden ulkopuolelle näkyvä viestien vaihto. Ulkoisella kuvauksella määritellään samalla palveluiden käyttäytyminen, joka on yksi tärkeistä yhteistoimintaan vaikuttavista tekijöistä. Ulkoisten kuvauksien avulla voidaan päätellä voiko kaksi palvelua toimia yhdessä vai ei [56]. Työnkulunkuvaus tai liiketoimintaprosessi myös määrittelee yksityiskohtaisesti sopimusosapuolten roolitukset, suhteet ja vastuualueet [19]. Näiden määritysten avulla kanssakäynti helpottuu oleellisesti.

Sopimuksen tulee sisältää kahdenlaisia velvoitteita [47]. Velvoitteiden tyypit ovat *mitä* pitää tehdä ja *milloin, miten ja missä* pitää tehdä. Näiden tyyppien avulla saadaan tarkasti määriteltyä tietyn organisaation kokonaisvastuu tietystä osatoiminnasta vaikka virtuaaliorganisaatiossa. Esimerkki voisi olla velvoite tuottaa lämpötilantarkastelu palvelu (*mitä*). Palveluun liittyen voidaan velvoittaa tuottamaan se sähköisesti, siten että se on käytettävissä internetin avulla ja sen pitää olla saatavilla sopimuksen voimassaoloajan 24 tuntia vuorokaudessa (*milloin, miten ja missä*).

Käyttäytymisen yhteyteen voidaan lisätä erillisiä virhetilanteiden käsittelyyn erikoistuneita kuvauksia. Tämänkaltainen ratkaisu mahdollistaa monimutkaisemmat käyttäytymiskuvaukset sopimusten toiminnan aikana ilmaantuvien poikkeustilantei-



Kuva 2.3: CPA:n kolme tasoa [11].

den ilmaisemiseen [13]. Kaikkia erilaisia virhetilanteita ei kuitenkaan voida ennustaa, joten jonkin kuvauksen tulisi olla geneerinen. Erilaiset virhetilanteet riippuvat myös käytettävistä toteutusalustoista. Sen vuoksi sopimuksen tulisi olla modulaarinen tässä suhteessa ja sallia erilaisten poikkeuskäsittelyiden lisääminen tarpeen mukaan.

Esimerkkitapauksessa virhetilanteiden käsittelystä olisi ollut hyötyä, jotta olisi havaittu heti, ettei Suomen Telkku OY pystynytkään tulkitsemaan Svensk Elektronik AB:stä lähetettyä laskua. Samoin olisi huomattu ongelma toimituksen saapumisilmoituksen yhteydessä. Virhetilanteiden käsittely edellä mainituissa tilanteissa nopeuttaa ongelman käsittelyä ja sen poistamista järjestelmistä.

Sopimusten päivittäminen on usein tarpeellista [13]. Sopimusta pitää muuttaa virtuaaliorganisaation muutosten yhteydessä. Muutoksia tapahtuu sekä jäsenten vaihtuessa että virtuaaliorganisaation rakenteen muuttuessa. Näiden tilanteiden varalta sopimuksessa tulee olla käsittelyohjeet. Esimerkkitilanteessa sopimusta jouduttiin muokkaamaan, kun aluksi ruotsalaisyrityksen valitsema kuljetusyritys ei pystynytkään liittymään Suomen Telkku OY:n tietojärjestelmiin.

Sopimukseen pitää kirjata palvelun laadulle asetetut kriteerit. Palvelun laatuun liittyvät määreet kuten aika ja kustannukset voidaan kuitenkin esittää vain epäsuorasti [24], joten työnkulkukuvauksia ei voida pakottaa toimimaan palvelun laadun mukaisesti. Ainoastaan rakenteellisia rajoitteita voidaan tehdä. Työnkulunhallintajärjestelmän (*workflow management system*, WfMS) hidastellessa voi siksi esiintyä ongelmia palvelun laadun suhteen. Kuitenkin tiettyjä palvelun laadun parametreja voidaan seurata. Näitä ovat suoritusaika ja keskeytettyjen aktiviteettien kustannukset.

ebXML:ssä pääsääntöiset sopimuselementit 'Resources and Contracts' metamallissa ovat *agreement* (yksimielisyys), *contract* (sopimus) ja *commitment* (sitoumus) [9]. Nämä edustavat erilaisia vaiheita sopimuksen muodostumisessa. Aluksi saavutetaan yksimielisyys sopimusten sisällöstä ja yksityiskohdista. Tämän jälkeen muodostetaan lopullinen sopimus perustuen yksimielisyyteen. Lopuksi sopimuksen osapuolten on vielä sitouduttava sopimukseen. ebXML:n sopimuksiin liittyvät olennaisena osana liiketoimintaprosessit ja -säännöt. Näiden avulla kuvataan palveluiden teknistä käyttäytymistä sopimuksia varten. Liiketoimintaprosessien avulla tarkkaillaan myös sopimuksessa olevien sitoumuksien täyttymistä.

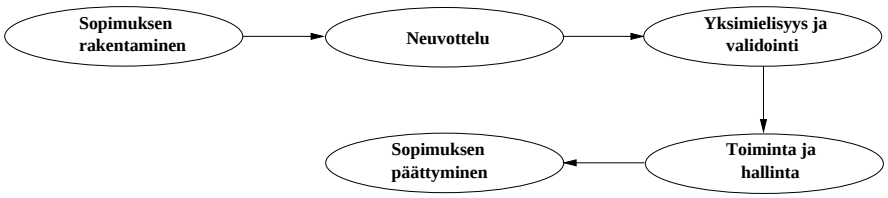
ebXML:n sopimusrakenteena on *Collaboration Protocol Agreement* (CPA). Tämän rakenteen kolme tasoa on esitelty kuvassa 2.3. Tasot ovat *Possibilities* (mahdollisuudet), *Capabilities* (kykeneväisyydet) ja lopulta *Agreements* (sopimukset, yhteisymmärrystilat). Mahdollisuudet kattavat kaikki erilaiset olemassa olevat erilaiset ratkaisut kahden osapuolen välillä muodostaen suurimman kokonaisuuden näistä kolmesta tasosta. Kykeneväisyydet rajoittavat mahdollisuuksia erilaisiin yhteisiin ratkaisuihin, joita molemmat pystyvät tukemaan. Tämä on suppeampi joukko kuin mahdollisuudet. ebXML:n sopimus on lopulta se yksikäsitteinen ratkaisu johon päädytään osapuolten välillä. Se määrittelee yhden alhaisen tason kommunikointitavan, yhden liiketoimintaprosessin ja niin edelleen. CPA:n pohjalla olevat XML-kuvaukset on rakennettu kahdelle osapuolelle joten se on aina kahdenvälinen sopimus. Kahdenväliset sopimukset eivät kuitenkaan riitä kaikkiin tilanteisiin.

OWL-S:llä ei ole varsinaista sopimusrakennetta. Yksinkertaisia sopimuksia voidaan rakentaa kuvaamalla eri sopimusosapuolten näkökulmasta niiden toimintaa. Toiminta sarjallistetaan joukoksi askelia (*steps*). Samalla, kun osapuolen käyttäytyminen kuvataan, tälle annetaan rooli ja rooliin liitetään käyttäytymiskuvaus. Lisäksi sopimukseen voidaan lisätä erillisiä rajoituksia, jotka ohjaavat käyttäytymistä sopimuksen alla. Jotta sopimus olisi tietokoneella ymmärrettävä, siihen lisätään vielä ontologia kuvaus esittämään sopimuksessa käytettyjä käsitteitä. Sopimuksesta saadaan helposti liiketoiminta-alueeseen sidottu, kun siinä käytetään liiketoiminta-aluekohtaista ontologiaa.

2.4 Sopimuksen elinkaari

Yritysten välisen (B2B) kommunikaation mahdollistavat teknologiat tarjoavat myös välineitä sopimusten tukemiseen [30]. Sopimusten ja niiden hallinnan tukeen tarvitaan seuraavia palveluita: sopimuksen esittäminen sähköisesti, osapuolten etsintä ja sopimuksen luominen, neuvottelupalvelut, sopimuksen validointi ja sopimuksen pykälien valvonta automaattisesti. Palveluiden tulee muodostaa saumaton kokonaisuus, joka tukee koko elinkaarta.

Sopimusten luomiseen tarvittavia palveluita ovat palvelurekisterit, kuten UDDI [38] ja CORBA Trader [42], joiden avulla osapuolet hakevat palveluita joko asiakaspalvelin liiketoimintaan tai virtuaaliorganisaatioihin osallistujiksi. Lisäksi tarvitaan väliohjelmistopalveluita, joiden avulla sopimuksia voidaan luoda löydettyjen osapuolten pohjalta. Yksi vaihtoehto on käyttää valmiita sopimuspohjia niitä tarjoavista varastoista tai luoda sopimuksia tyhjästä. Sähköisen esityksen yhtenäisyyden vuoksi valmiiden sopimuspohjien käyttäminen on suositeltavaa. Neuvottelun avulla pyritään pääsemään yhteisymmärrykseen sopimuksen ehdoista ja yksityiskohdista. Sopimuksen validoinnin yhteydessä varmistetaan sen yleinen järkevyys ja ettei se ei ole liiketoiminta-alueella voimassaolevien lakien vastainen. Sopimuksen valvonnan aikana pyritään havaitsemaan sopimuksen vastainen toiminta ja puuttumaan siihen. Neuvottelusta ja valvonnasta erikseen seuraavissa kappaleissa.



Kuva 2.4: Sopimuksen elinkaari yksinkertaistettuna.

Kuvassa 2.4 on esitetty yksinkertaistettu elinkaari sopimukselle. Elinkaari koostuu viidestä vaiheesta, jotka ovat rakentaminen, neuvottelu, yksimielisyyden saavuttaminen sopimuksen yksityiskohdista ja validointi, toiminta ja hallinta sopimuksen aikana ja sopimuksen päättyminen [34]. Rakentamisen aikana valitaan sopimuspohja ja etsitään sopimuspohjaan muut osallistujat. Neuvottelun jälkeen on erillinen validointi yksimielisyyden saavuttamisen jälkeen. Toiminnan ja hallinan aikana sopimuksen alaisia toimintoja valvotaan ja huolehditaan siitä, että sopimuksen alaiset palvelut pysyvät käytettävissä. Myös mahdolliset osapuolten vaihdot kuuluvat hallinnan piiriin. Sopimuksen umpeutuessa sopimukseen liittyvä virtuaaliorganisaatio puretaan tai ainakin palveluiden käyttäminen vanhan sopimuksen alaisesti estetään.

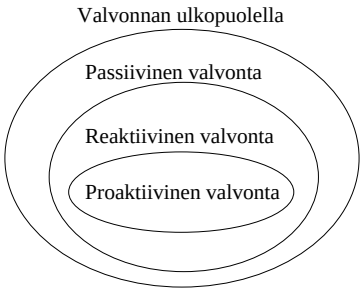
Neuvottelu käydään kaikkien sopimusosapuolten välillä ja sillä pyritään mahdollisimman hyvään ratkaisuun kaikkien osapuolten kannalta. Neuvottelu on siis optimointiongelma useiden erilaisten tavoitteiden suhteen. Neuvotteluja käydään virtuaaliorganisaation luonnin aikana ja sen jälkeen aina, kun osapuolten välillä on erimielisyyksiä tai epäyhteensopivuuksia [47].

Neuvottelut alkavat aina jonkun aloitteesta [4]. Tyypillinen esimerkki tästä on asiakas-palvelin-suhde. Tämä tarkoittaa sitä, että jonkun osapuolista on oltava vetovastuussa neuvotteluissa. Vetovastuussa oleva on velvollinen edistämään neuvotteluja ja muokkaamaan sopimusta neuvottelujen edetessä tarvittavilla muutoksilla. Vetovastuullista pitää myös voida vaihtaa, jos kyseessä oleva osapuoli päättää lopettaa neuvottelut itselleen tuloksettomina.

Ongelmana automatisoiduissa neuvotteluissa on se, ettei tyydyttävää geneeristä neuvottelumallia ole olemassa [4]. Tämä on ongelma väliohjelmistoilla ja hajautetuilla sovelluksilla. Tarvittavan neuvottelumallin pitää olla luotettava, jotta sitä voidaan käyttää. Niinpä neuvottelun onkin tapahduttava tietokoneavusteisesti. Tämä tarkoittaa sitä, että ihmisten on hyväksyttävä neuvottelutulokset ja tehtävä ehdotuksia neuvottelukierroksilla. Tietokoneet voivat kyllä avustaa ihmisiä neuvotteluissa ja tehdä ehdotuksia ihmisten käytettäväksi.

Sopimuksen toimeenpano vaihtelee yrityksestä toiseen riippuen niiden valmiuksista ja teknisistä lähtökohdista [17]. Toimeenpanon valmiudet vaikuttavat siihen, miten hyvin sopimuksen noudattamista kyetään valvomaan ja sopimusrikkomuksista raportoimaan. Lopulta kuitenkin sopimuksen toimeenpano on ihmisten käsissä. Kaikkia sopimusrikkomuksia ei pystytä havaitsemaan automaattisesti.

Sopimuksen toimeenpano voidaan tehdä kolmella erilaisella tavalla. Nämä ovat proaktiivinen, reaktiivinen tai passiivinen [33]. Proaktiivisesti tehdyllä toimeenpa-



Kuva 2.5: Valvonnan kolme tasoa.

nolla pyritään aktiivisesti estämään sopimusrikkomukset ennen niiden tapahtumista. Reaktiivisuus tarkoittaa sitä että pyritään reagoimaan sopimusrikkomuksiin välittömästi niiden tapahduttua ja korjaamaan tilanne. Passiivinen toimeenpano tarkoittaa toiminnasta syntyneiden lokitietojen tarkistamista jälkikäteen ja päättelyä niiden pohjalta menikö kaikki sopimuksen mukaan vai ei. Näillä erilaisilla tavoilla voidaan havaita vain tietty määrä erilaisia rikkomuksia. Eri tavoilla havaittavien rikkomusten lukumäärää on kuvattu kuvassa 2.5. Proaktiivisesti voidaan havaita vähiten rikkomuksia ja passiivisesti kaikkein eniten rikkomuksia. Kaikkia rikkomuksia ei välttämättä voida havaita.

Toimeenpanoa hoidetaan muunmuassa valvomalla kommunikointia. Kommunikoinnin valvonnan avulla tiedetään mitä tapahtuu, voidaan havaita sopimusrikot ja seurata kumppaneiden toimintaa [47]. Valvonnan avulla voidaan seurata sekä omaa toimintaa, että muiden osapuolten toimintaa. Valvonnalla voidaan varmistaa sekä oman yrityksen palveluiden että muiden yritysten palveluiden käyttäytymisen sopimuksen mukaisuus [33]. Valvonnasta saadut lokitiedot ovat hyödyksi mahdollisissa kiistoissa sopimuksen rikkomisesta. Niitä voidaan käyttää sekä puolustautumiseen muiden väitteiltä rikkomisesta että tukena osoitettaessa, että joku muu on rikkonut sopimusta.

Sopimusrikkomusten tapauksessa tarvitaan erillistä neuvottelupalvelua [47]. Sen avulla päästään yhteisymmärrykseen tilanteesta ja voidaan joko jatkaa toimintaa tai purkaa sopimus. Riippuen rikkomusten vakavuudesta voidaan joko neuvotella suoraan yritysten välillä tai käyttää lakiin pohjautuvia palveluita kuten oikeustoimia tai välimieskäsittelyä. Neuvottelupalvelun ei siis tarvitse olla sähköisesti toteutettu ollakseen käyttökelpoinen.

Palvelun laatu on yksi valvottavista asioista sopimuksen voimassaolon aikanan. Osa palvelun laadun parametreistä on suoraan valvottavissa [24]. Valvottaviksi parametreiksi lasketaan suoritusaika ja keskeytettyjen aktiviteettien kustannukset. Edellisistä yhdistämällä saadaan joukko erilaisia johdettuja palvelun laadun kriteerejä.

Sopimuksen validointi syntymisen jälkeen varmistaa sen, että sopimus on järkevä ja ettei se ole ristiriidassa lain suhteen. Validointi onkin suositeltavaa [33]. Validoinnin lisäksi sopimuksen toimintakuvaukset tulee verifioida, jotta teknisissä kuvauksissa ei ole ristiriitaisuuksia. Koneellisella verifioinnilla ei, kuten valvonnallakaan, voida havaita kaikkia ongelmia palveluiden teknisessä yhteistoiminnassa. Verifiointi on si-

tä hankalampaa mitä monimutkaisempia käyttäytymiskuvaukset ovat. Palveluiden välillä mahdollisesti olevat semanttiset erot käsitteiden merkityksissä vaikeuttavat verifiointia. Verifioitu sopimus on kuitenkin edellytys valvonnan ja samalla toimeenpanon käyttökelpoisuudelle.

Sähköiset allekirjoitukset lisäävät tietoliikenteen varmuutta toiminnan aikana [17]. Sähköisiä allekirjoituksia on vaikea väärentää, joten niiden avulla voidaan varmistaa viestien alkuperä ja varmentua siitä, että ne todella on lähetetty kyseessä olevan yrityksen toimesta. Sopimuksen osallistujat voidaan myös todentaa allekirjoitusten avulla ja siten varmistaa osallistujien sitoutuminen sopimukseen.

ebXML:n sopimuksissa käytetään liiketoimintaprosessien kuvauksia, joiden suoritusta voidaan valvoa. Kuvausten avulla ebXML:n sopimuksia voidaan valvoa. OWL-S pohjaisten sopimusten toimeenpano ja valvonta vaatii tekniseltä ympäristöltä päättelykykyä. Päättelykykyä tarvitaan sopimuksissa olevien kuvausten tulkintaan ja kuvausten avulla tapahtuvaan valvontaan.

ebXML määrittelee hakemistopalvelun, jonka avulla voidaan sopimuksen rakennusvaiheessa etsiä sopivia palveluita. Tämän palvelun nimi on ebXML Registry Service (ebRS)[4]. Palvelun avulla voivat palvelun tarvitsijat etsiä palveluita ja palvelun tuottajat mainostaa palveluitaan. ebRS:n avulla voi hakea ainoastaan yksittäisiä palveluita eikä sillä voi varmistaa palvelujoukon yhteentoimivuutta. OWL-S ei varsinaisesti tarjoa palveluita sopimuksen elinkaaren hallintaan.

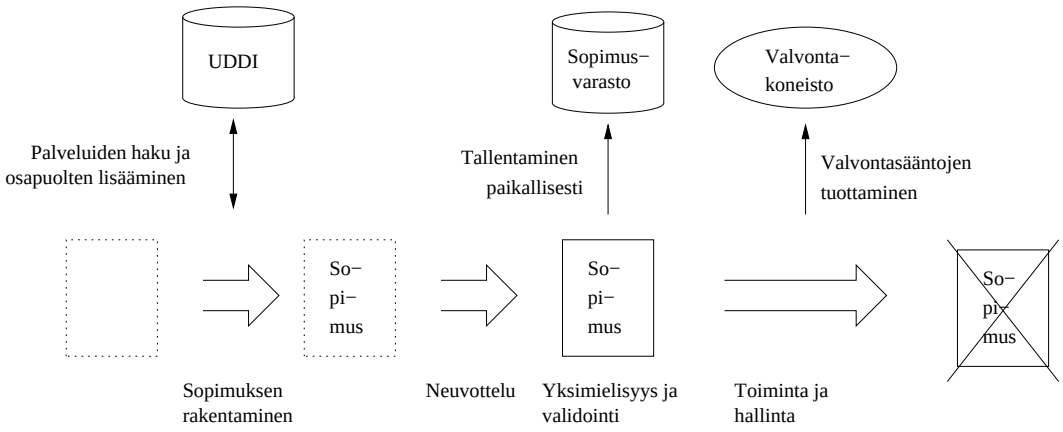
2.5 Tekninen ympäristö

Pelkällä sähköisesti esitetyllä sopimuksella ei päästä kovinkaan pitkälle koko prosessin sähköistämisessä. Tämän vuoksi sopimusten ympärille tarvitaan tekninen ympäristö, jonka avulla sopimuksia, niiden sisältöä ja elinkaarta pystytään hallitsemaan.

Tekniselle ympäristölle voidaan määritellä viisi tärkeää vaatimusta. Nämä ovat sopimuksen elinkaaren hallinta, sopimuskieli sopimuksen ilmaisemiseeen, turvallisuus/ luottamuksellisuus, integroitavuus olemassa oleviin järjestelmiin ja alustariippumattomuus [34]. Näistä vaatimuksista vaikeimmin saavutettavia ovat integrointi olemassa oleviin järjestelmiin ja sopimuskielen rakentaminen. Sopimuskielen ongelmana on lakiin pohjautuvien sopimusten mallintaminen ja integroinnin ongelma on laaja määrä erilaisia olemassa olevia järjestelmiä, joihin integrointi pitäisi pystyä tekemään.

Elinkaaren hallintaan liittyvät palvelut tarkoittavat luvussa 2.4 eritellyn sopimuksen elinkaaren vaiheisiin liittyviä palveluita. Palvelut ja sopimuksen elinkaari on esitetty kuvassa 2.6. Sopimuksen muodostusvaiheessa palveluhakemistosta haetaan sopivia palveluita ja ne lisätään käytössäolevaan sopimuspohjaan. Osapuolten löytymisen jälkeen sopimus neuvotellaan tietokoneavusteisesti ja tallennetaan paikallisesti varastoon. Sopimuksen käytön aikana siitä tuotetaan valvontasäännöt monitoreille ja sopimuksen muuttumisen yhteydessä neuvotellaan uudestaan sopimuksen yksityiskohdista. Sopimuksen umpeuduttua järjestelmä purkaa virtuaaliorgani-

[4]http://www.ebxml.org/specs/ebrs2.pdf



Kuva 2.6: Elinkaaren hallintaan liittyviä palveluita.

saation ja ajaa organisaatioon liittyvät palvelut alas. Järjestelmän tulee siis tarjota välineet elinkaaren hallintaan tai vaihtoehtoisesti itsensähallitseva elinkaari siten, että elinkaaren vaiheet etenevät loogisesti ja eivät vaihdu ennen kuin se on tarkoituksenmukaista.

Sopimuskielellä tarkoitetaan tapaa ilmaista sopimus ja siihen liittyvät asiat. Kielen pitäisi olla sellainen, että sillä tehtyjä sopimuksia voidaan valvoa. Kielen avulla pitää voida esittää kaikki aikaisemmin esitellyt vaatimukset sopimusten sisällölle. Sisällön lisäksi ei pidä unohtaa liiketoiminta-alueen lakeja ja muita säädöksiä. Kielen avulla rakennettujen sopimusten pitäisi siis olla validoitavissa siten, että voidaan varmistaa niiden laillisuus ja järkevyys. Laskennallinen verifiointi auttaa varmistamaan, että sopimusten perusteella yhteistoimintaan pyrkivät sovellukset todella siihen pystyvät teknisesti. Tämä tarkoittaa sitä ettei esimerkiksi kommunikointiprotokollissa ole lukkiutumia.

Kuten sopimusten sisällönkin kohdalla myös teknisen ympäristön tulee pystyä sopimuspohjien käyttämiseen sopimusten rakentamisen aikana. Tuen pitää ulottua standardoitujen osien käyttämiseen sopimusten pohjana ja lisäosina [9]. Tämä helpottaa sopimuksen laillisuuden varmennuksessa. Verifiointi on raskasta ja sitä ei suositella tehtäväksi ajonaikaisesti sopimuksen rakennuksen aikana [28]. Tämän vuoksi se pitää tehdä etukäteen heti sopimuksen tai toimintakuvausten mallinnuksen jälkeen. Tosin verifionti voi olla käytännössä tehokasta. Etukäteen suoritettu validointi ja verifiointi lyhentää sopimuksen rakennukseen käytettyä aikaa suhteessa ajonaikaiseen validointiin ja verifiointiin.

Koska yritysten välinen sähköinen liiketoiminta tapahtuu Internetissä, pitää teknisen ympäristön tukea Internet-pohjaisia ratkaisuja sopimusten hallintaan [17]. Tämä tarkoittaa käytännössä XML-pohjaisten kielien tukemista ainakin sopimuksen tai sen osien siirtämisessä organisaatioista toiseen. Lisäksi järjestelmän tulee tukea sähköisiä allekirjoituksia. Allekirjoitusten avulla nostetaan käyttäjien luottamusta järjestelmään ja parannetaan sen turvallisuutta. Allekirjoitukset parantavat luottamusta, koska ne toimivat hyvin varmana todennusjärjestelmänä. Myös turvallisuus

paranee varman todentaminsen vuoksi ja mahdollistaa salauksen käyttämisen yritysten väliseen kommunikointiin.

Sopimuksissa tulisi käyttää yleistä termistöä, ja teknisen ympäristön tehtävänä on muuntaa yleinen termistö yrityksen sisäiseksi termistöksi. Tällä vähennetään virtuaaliorganisaatioiden osallistujien riippuvuutta toisistaan ja sopimuksen termistöstä [29]. Termistön muuntamisella paikalliseen käyttöön edistetään edellä mainittua integroitavuutta, mutta samalla monimutkaistetaan sopimusten hallintaan erikoistuneen ympäristön toimintaa. Tämän vuoksi teknisen ympäristön tulisikin olla hyvin modulaarinen, jotta sopimuksen hallintaympäristö voidaan konfiguroida sopimaan yhteen aikaisempien järjestelmien kanssa.

Termistön muuntamisesta huolimatta sopimukseen ja virtuaaliorganisaatioon osallistuvien partnereiden pitää semanttisella tasolla jakaa yksiselitteinen näkemys geneeristen dokumenttien [19, 51]. Vaikka sopimus ja sen osat olisikin validoitu ja todettu yhteentoimiviksi, muunnoksella tuotettavat semantiikkaan liittyvät erot osapuolten välillä voivat johtaa epäonnistumiseen. Tämä pätee varsinkin yritysten välillä liikutettaviin dokumentteihin. Jos dokumentteina käytetään valmiiksi validoituja standardoituja dokumentteja, niiden kenttien semantiikka pitää varmistaa sopimalla siitä sopimuksessa. Esimerkiksi virtuaaliorganisaatiossa käytettävä valuuttayksikkö pitää sopia.

Semanttisen yhtenäisyyden varmistamiseksi sopimuksen käsitteistä ja niiden suhteista voidaan muodostaa ontologia. IT-järjestelmissä ontologioita käytetään usein kuvaamaan tietoa erilaisista konsepteista ja niiden suhteista [50]. Ontologioiden avulla voidaan myös yhdistää useita erilaisia liiketoiminta-alueita ja niiden erikoistuneita sopimuksia. Jotta aiemmin mainittu termistön muuntaminen yleisestä sisäiseksi olisi käytännössä mahdollista koneellisesti, niiden väliin tarvitaan myös ontologia.

Sopimuksia on myös kyettävä muokkaamaan niiden solmimisen jälkeen. Sopimusten päivittäminen on usein tarpeellista [13]. Usein vanhoja sopimuksia käytetään pohjana uusien sopimusten tekemisessä. Vanhasta sopimuksesta otettu pohja voidaan sitten täyttää uudestaan. Myös erilaiset muutokset virtuaaliorganisaation rakenteessa saattavat vaativat sopimuksen päivittämistä. Rakenteen muutokset voivat aiheuttaa osallistujien määrän lisääntymisen, vähenemisen tai kommunikointiin vaikuttavan muutoksen esimerkiksi kahden palvelun yhtenäistämisen ja uuden luomisen yhteydessä.

Turvallisuus ja luottamus vaikuttavat järjestelmän suunnitteluun. Turvallisuus tarkoittaa sitä että järjestelmän suojaukset ja kommunikaation salaus ovat riittävällä tasolla, jotta tarpeettomia informaatiovuotoja ei pääse tapahtumaan. Mahdolliset vuodot saattavat aiheuttaa suuria menetyksiä virtuaaliorganisaatioihin osallistuville yrityksille. Järjestelmän ei tulisi luovuttaa tarpeettomia tietoja yrityksestä ulos edes virtuaaliorganisaation sisällä [33]. Tämä johtuu siitä, että usein yritysten sisäiset politiikat ovat luottamuksellisia. Tosin yrityksen sisäiset politiikat näkyvät epäsuorasti ulos niiden käyttäytymisen välityksellä. Luottamuksella järjestelmään tarkoitetaan käyttäjien luottamusta sen toimintaan. Jotta käyttäjien luottamus voitetaan, on järjestelmän tuotettava sellaisia tuloksia kuin käyttäjät siltä odottavat.

Käyttäjällä tarkoitetaan tässä tilanteessa organisaation sopimuksista vastaavaa ihmistä. Järjestelmän on myös oltava tarpeeksi luotettava ja käytettävyyden on oltava tarpeeksi korkealla tasolla.

Sopimustenhallintajärjestelmän integrointi olemassaoleviin järjestelmiin on haasteellinen vaatimus. Jo se että mahdollisia järjestelmiä, joihin voidaan integroida, on useita, vaikeuttaa integrointia. Integrointi pitäisi myös suorittaa olemassa olevia järjestelmiä suuremmin muuttamatta. Jos järjestelmiä tarvitsee muuttaa, kustannukset voivat nousta suuriksi. Käytännössä sopimuksenhallintajärjestelmä pitäisi kaikkine ominaisuuksineen voida lisätä olemassaolevan järjestelmän rinnalle siten, että mahdollisimman pienin muutoksin kaikki olemassaolevat järjestelmät edelleen toimivat.

Sopimus ja hallintaympäristö tulee toteuttaa mahdollisimman riippumattomaksi valitusta hajautustekniikasta ja toteutusalustasta. Ominaisuuksia tulee standardoida siten, että useita erilaisia toteutuskieliä on mahdollista käyttää. Ongelmia erilaisista alustoista on keskinäisen kommunikoinnin yhteydessä. Kuitenkin voidaan valita tekniikoita, jotka edesauttavat alustariippumattomuutta. Esimerkki tällaisesta tekniikasta on XML, jota voidaan lukea ja tulkita useilla erilaisilla alustoilla.

Teknisen ympäristön kannalta sopimus muodostaa kontekstin. Konteksti voidaan määritellä seuraavasti: kontekstitietoa voi olla mikä tahansa tieto, jota voidaan käyttää esimerkiksi olion tilan kuvaamiseen liittyen johonkin ominaisuuteen [50]. Sopimuksen hallintaympäristön kannalta erityisesti valvonta on sellainen ominaisuus, joka on riippuvainen sopimuksesta eli kontekstista. Valvonnan on, valvoessaan sopimusta, tiedettävä mistä sopimuksesta on kulloinkin kyse. Sopimustenhallintajärjestelmä on kontekstitietoinen, jos se osaa muokata käyttäytymistään sopimuskohtaisesti.

Luku 3

Sopimuksen tietosisältö

Sopimuksen tietosisältöön kuuluu joukko kahden tai useamman palvelun yhteistoiminnan varmistamiseen liittyvää tietoa. Lisäksi sopimuksiin kuuluu tietoa sopimusten hallintaan osallistuvaa tietojärjestelmää varten. Yhteistoiminnan varmistamiseksi palveluiden käyttäytyminen ja tarpeet tulee kuvata sopimuksessa. Sopimuksessa olevan tiedon määrä riippuu osapuolten määrästä. Osapuolten määrä lisää palveluiden määrää ja palveluiden määrä lisää käyttäytymiskuvausten määrää. Toisin sanoen sopimus toimii eräänlaisena tietovarastona, jonka perusteella osapuolet osaavat muokata teknisen ympäristönsä asetukset sellaiseksi, että yhteistoiminta onnistuu tarkoitetulla tavalla.

Sopimuksen tietosisältö voidaan luokitella kolmeen luokkaan: toiminnallisiin ominaisuuksiin, ei-toiminnallisiin ominaisuuksiin ja sopimuksen hallintaan liittyviin teknisiin osiin. Toiminnalliset ominaisuudet kertovat palveluiden käyttäytymisestä ja yleisestä toiminnasta. Näihin ominaisuuksiin voidaan laskea palveluiden käyttäytymiskuvaukset ja tietoliikenteelle asetetut vaatimukset. Ei-toiminnalliset ominaisuudet antavat toiminnallisille ominaisuuksille lisämääreitä, joita toiminnan aikana tulee noudattaa. Esimerkkejä näistä ominaisuuksista ovat erilaiset luottamukseen liittyvät tekijät ja tosiaikavaatimukset. Sopimuksen hallintaan liittyvät ominaisuudet koostuva lähinnä erilaisista tunnisteista, joiden avulla tekninen hallintaympäristö erottaa sopimukset toisistaan.

Sopimuksen tietosisältö käsitellään seuraavasti. Aluksi esitellään sopimustenhallintaan liittyvät ominaisuudet, jotka ovat sopimusten tunnisteet ja voimassaolo sekä sopimusosapuolten informaatio. Seuraavaksi siirrytään käsittelemään palveluiden toimintakuvauksia ja palveluiden välistä kommunikaatiota. Lopuksi esitellään joukko ei-toiminnallisia ominaisuuksia, jotka voidaan jälkikäteen lisätä toimintakuvauksiin ja tietoliikenteeseen lisämääreinä.

3.1 Sopimuksen tunnistaminen ja voimassaolo

Sopimuksella pitää olla tunnisteet, jotta se voidaan tunnistaa sekä tietojärjestelmiin talletuksen vuoksi että siihen liittyvän kommunikaation vuoksi. Sopimuksen voimas-

Nimi	Alku	Loppu
Absoluuttinen	31.5.2004	31.12.2004
Suhteellinen (1)	31.5.2004	vuoden päästä
Suhteellinen (2)	heti	vuoden päästä
Istuntoihin sidottu	31.5.2004	100 istuntoa

Taulukko 3.1: Erilaisia mahdollisia voimassaolon määrittämisen tapoja.

saolo on taas merkittävää sitoumuksen pituuden ja määrän rajaamiseksi haluttuihin mittoihin. Istuntojen avulla tuetaan useaa samanaikaista toiminnallisuuden suorituspolkua.

Tunnisteet tarvitaan sekä sopimukselle itselleen, että sopimuksen alaisille istunnoille. Sopimuksen tunniste tarvitaan ensimmäisen sopimusehdotuksen jälkeen tai silloin, kun sopimus on valmis tallennettavaksi paikalliseen varastoon. Jos sopimuksen muodostamiseen käytetään teknisiä palveluita, tunniste tarvitaan ensimmäisen ehdotuksen aikana. Koska sopimukseen sisältyy useita osapuolia mahdollisesti eri puolilta maailmaa, pitäisi sopimuksen tunnisteen olla maailmanlaajuisesti ainutlaatuinen (*globally unique*). Siten se ei voi olla sama yhdenkään osallistujan aikaisempien sopimusten tunnisteiden kanssa.

Maailmanlaajuisesti ainutlaatuisten tunnisteiden ongelmana on niiden luominen. Yksi ratkaisu on keskitetty sopimusten tunnisteiden määrääjä, jolta kaikki hakevat tunnisteen ennen sopimuksen muodostamista. Tämä on kuitenkin mahdoton ratkaisu suorituskykyyn ja saatavuuteen (*availability*) liittyvien ongelmien takia. Toinen mahdollisuus on käyttää satunnaista tunnistetta. Tämän tunnisteen on oltava riittävän pitkä, jotta todennäköisyys tunnisteiden yhteentörmäykselle on riittävän pieni. Tunnisteen yhteentörmäyksellä tarkoitetaan tilannetta, jossa kahdella sopimuksella on sama tunniste. 160 bitin mittainen tunniste alkaa olla riittävä. Tällöin yhteentörmäyksen mahdollisuus on hyvin pieni (erilaisia mahdollisuuksia on $1,46 * 10^{48}$). Tämä ratkaisu vaatii satunnaisalgoritmia, jonka jakauma on tasainen. Kolmas mahdollisuus on käyttää apuna meklauspalvelua [25] (*trading service*) osapuolten löytämiseen ja samalla pyytää meklauspalvelua luomaan sopiva tunniste. Tämä kuitenkin vaatii meklauspalvelua, joka osaa palauttaa joukon osapuolia sopimukseen ja luoda joukolla yksikäsitteisen tunnisteen.

Jokaisella istunnolla on tila. Istunnon tilalla tarkoitetaan sopimuksessa sovitun toiminnallisuuden käytön etenemisastetta. Sopimuksessa määriteltyyn toiminnallisuuteen liittyvät tapahtumat sidotaan siis aina johonkin tiettyyn istuntoon. Tämä mahdollistaa myös sopimuksen pohjalta tehdyn valvonnan istuntokohtaisesti, jolloin eri istuntojen viestit sekoitu keskenään.

Sopimuksella tulee olla voimassaoloaika. Erilaisia voimassaolon määrittelytapoja on esitelty taulukossa 3.1. Sopimuksen voimassaolo voidaan määritellä asettamalla alku- ja päättymispäivämäärät. Tässä tapauksessa voimassaolo on kaikkein selkeimmin ilmaistu. Vaihtoehtoina on kuitenkin ilmaista alkamisaika päivämääränä ja sitten antaa suhteellinen päättymispäivämäärä (esimerkiksi yksi vuosi) ja ilmoittaa

ensin alkupäivämäärä ja sen jälkeen sallittujen istuntojen määrä. Edellinen tapaus voidaan aina muuttaa ensimmäiseksi ja tietokoneilla käsiteltynä usein muunnetaankin. Jälkimmäinen mahdollisuus ei siten varsinaisesti rajoita sopimuksen käyttöaikaa, mutta antaa selkeän sopimuksen päättymistavan ja ajankohdan. Tämä on hyödyllistä, jos halutaan sopia esimerkiksi viidestä toimituserästä, joiden lopullinen tilaus ja toimitusaika riippuu tilaajan jälleenmyynnin määristä. Sopimus umpeutuu viidennen toimituksen jälkeen ja tilaaja on vapaa hankkimaan uuden toimittajan tai neuvottelemaan uuden sopimuksen vanhan jatkoksi.

3.2 Osallistujien informaatio

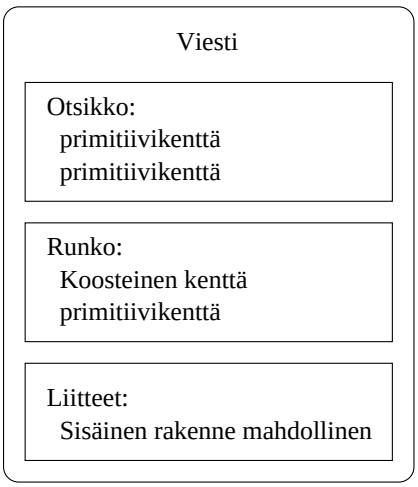
Osallistujien tunnistaminen on tarpeellista, kun sopimusta hallitaan ohjelmallisesti. Tunnistusta tarvitaan palvelun käyttäjän yksilöimiseksi ja siten valvonnan helpottamiseksi. Jotta sopimuksenalaisessa toiminnassa voidaan ratkaista vastuukysymyksiä ja tekojen seurauksia, tarvitaan myös luotettavaa osapuolten tunnistamista. Osapuolen tunnisteen ongelmina on yksikäsitteisyys ja tunnisteen rakentaminen sellaiseksi, ettei sitä voida väärentää.

Sähköiset allekirjoitukset on yksi keino tunnistaa osapuoli yksikäsitteisesti. Allekirjoitusten hyvänä puolena on myös se, että niiden väärentäminen on vaikeaa, joten ne takaavat osapuolen aitouden. Sähköiset allekirjoitukset eivät kuitenkaan ole täydellisen varma ratkaisu. XML-pohjaisiin sopimuksiin on luontevinta käyttää XML-pohjaista sähköistä allekirjoitusta. W3C on kehittänyt standardin sähköisille XML-allekirjoituksille [5].

Tunnisteiden lisäksi sopimuksessa on hyvä olla yleisempää tietoa yrityksestä kuten sen yhteystiedot. Lisäksi sopimuksesta vastuullinen henkilö pitää mainita sopimuksessa. Tämän yleisemmän tiedon tarkoituksena on tarjota mahdollisuus ihmisten väliselle yhteyden pidolle sopimuksen suhteen. Tästä on hyötyä esimerkiksi tilanteissa, joissa on rikottu sopimusta. Sopimusrikkoa ei välttämättä voi aina ratkaista suoraan tietokoneella, varsinkaan täysin automaattisesti. Tilanteessa, jossa osapuoli A syyttää osapuolta B rikkomuksesta ja osapuoli B ei myönnä tehneensä rikkomusta, joudutaan jo käyttämään apuna lakimiehiä. Lisäksi sopimuksessa tarvitaan vielä osoite osallistujan tarjoamiin automaattisiin sopimuksen muodostaman verkoston hallintapalveluihin mikäli sellaisia on.

Sopimuksessa pitää olla tekniset osoitteet jokaisen osapuolen tarjoamiin palveluihin. Tietojärjestelmät voidaan siten konfiguroida ottamaan yhteyttä sopimuksenalaisiin palveluihin. Palveluiden käyttäminen on mahdotonta, jos niiden osoitteita ei tiedetä. Web-palveluympäristössä osoitteet on esitetty URL:na[1].

[1]Universal Resource Locator, esimerkiksi http://www.cs.helsinki.fi/group/web-pil/



Kuva 3.1: Viestin rakenne.

3.3 Toimintakuvaukset

Toimintakuvaukset ovat tärkein osa sopimusta sen toimeenpanon kannalta. Ne kuvaavat sopimuksen osapuolen ulkoista käyttäytymistä siten, kun se näkyy muille sopimuksessa mukana oleville osapuolille. Olennaisena osana toimintakuvauksia ovat niissä käytetyt viestit (web-palveluiden tapauksessa) tai funktiokutsut (perinteinen hajautettu järjestelmä). Toimintakuvausten avulla sopimus voidaan etukäteen verifioida ja sopimusta voidaan ajonaikaisesti valvoa. Verifiointi on tärkeää, jotta voidaan etukäteen havaita mahdollisia ongelmia yhteentoiminnassa. Koska kaikkia ongelmia ei kuitenkaan voida etukäteen havaita, pitää vielä ajonaikaisesti valvoa, että kaikki toimii kuten pitääkin.

Yksinkertaisimmillaan toimintakuvaukset kertovat vain mahdolliset operaatiot ja niihin liittyvät parametrit tyyppeineen. Tämäntasoinen kuvaus ei kuitenkaan riitä kovinkaan pitkälle yhteistoiminnan varmistamisessa kuten myöhemmin havaitaan. Seuraava monimutkaisempi kuvaus on kuvata operaatioiden lisäksi niiden käyttämiseen liittyvä järjestys eli sovellusprotokolla. Protokollan perusteella voidaan jo paljon paremmin varmistaa kahden järjestelmän yhteentoiminta, mutta sekään ei riitä ihan kaikkeen. Viimeinen ja kaikkein monimutkaisin tapa on lisätä protokollaan semanttiset kuvaukset operaatioiden toiminnalle ja niiden parametreille. Tämä riittää jo kuvaamaan suurimman osan yhteistoimintaan vaadittavista tiedoista. Lopullinen yhteistoiminnan onnistuminen voidaan kuitenkin havaita vasta ajonaikaisesti, sillä palveluiden käyttäytyminen ei välttämättä vastaa teknistä kuvausta.

Toimintakuvauksiin liittyvät yritysten välillä lähetettävät viestit. Viestin rakenne on esitetty kuvassa 3.1. Viesti sisältää joukon otsikkokenttiä, rungon ja liitteet. Otsikkokentät sisältävät viestille lisämääreitä. Rungossa voi olla sekä primitiivityyppejä[2] että koosteisia tyyppejä. Liitteillä voi olla sisäinen rakenne.

[2] Java™-kielen tarjoamia primitiivityyppejä ovat kaksi erilaista liukulukua (double ja float), kaksi erilaista kokonaislukua (int ja long), totuusarvo (boolean) ja joukko muita (byte ja char).

```
<wsdl:message name="tilaaRequest">
   <wsdl:part name="tilaus" type="xsd:string"/>
   <wsdl:part name="federaatioID" type="xsd:string"/>
</wsdl:message>
```

Kuva 3.2: Viesti esimerkkitilanteesta kuvattuna WSDL:llä.

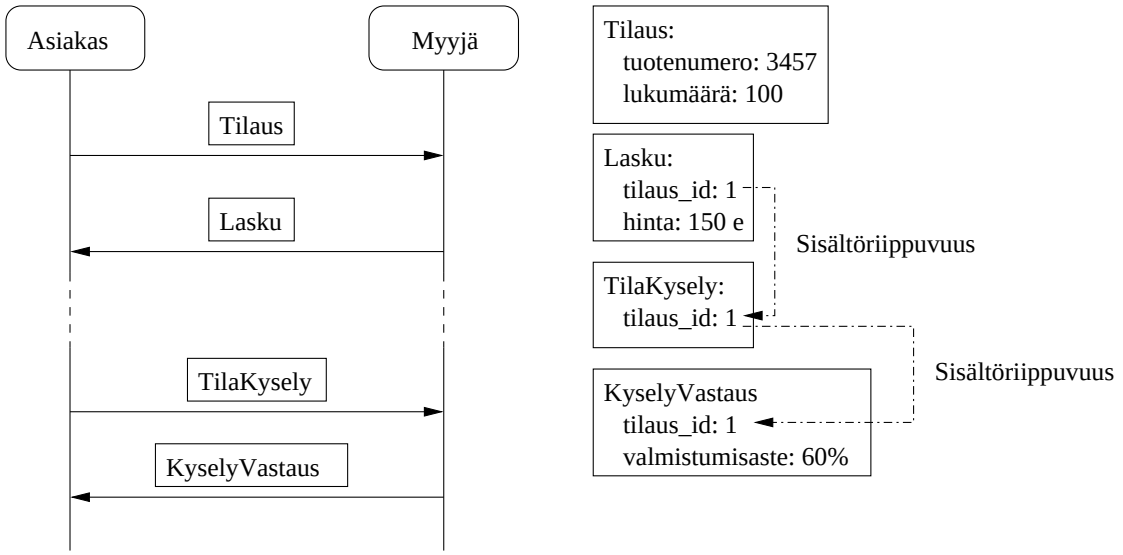
Viestit voivat kuulua kolmeen eri ryhmään: perusviestit, ilmoitukset (*notification*) ja virheilmoitukset (*error*). Perusviestit sisältävät liiketoimintaan liittyvää tietoa ja siirtävät toimintakuvauksen muodostamaa tilakonetta eteenpäin. Ilmoituksia käytetään esimerkiksi yksittäisten tapahtumien valmistumisen ilmoittamiseen. Ilmoituksen syy voi olla esimerkiksi postipaketin saapuminen lähimpään postiin. Virheilmoituksia käytetään erilaisten ongelmatilanteiden ilmaisemiseen. Virheen syynä voi olla esimerkiksi viestin sisällön tuhoutuminen (*corruption*) tai virheellinen viestityyppi.

Virhetilanteen aikana toimintakuvauksen suoritus keskeytyy. Virheitä on kahta tyyppiä: itsestään ratkeava väliaikainen virhe, joka poistuu yrittämällä uudelleen, ja pitkäaikainen virhe, joka johtuu isommasta ongelmasta kuten palvelun vääräst ä konfiguroinnista. Tämän vuoksi virheilmoitustyyppejä pitäisi olla kaksi erilaista edellämainittuja tilanteita varten.

Kuvassa 3.2 on kuvattu viesti luvun 2.2 esimerkkitilanteesta. Viestin tyyppi on *tilaaRequest* ja siinä on kaksi kenttää: tilaus ja federaatioID. Molempien kenttien tyyppi on merkkijono. Tilaus on myös koosteinen tyyppi muodostaen tilauksen yksityiskohdat. Koska viesti on määritelty XML-kielellä, merkkijonoon voidaan helposti tallentaa koosteinen tyyppi. Tilausta ei ole tämän esimerkin puitteissa määritelty sen tarkemmin.

Viestien lisäksi toimintakuvaukset määrittelevät viesteille järjestyksen, lähettäjän ja vastaanottajan. Tätä järjestystä kutsutaan viestiprotokollaksi. Protokolla määrää milloin tietty viesti pitää lähettää tai vastaanottaa. Koska sopimuksessa voi olla kaksi tai useampia osallistujia, pitää kuvauksissa ilmoittaa kuka viestin lähettää ja kuka on sen vastaanottaja. Protokollan määritteleminen on tärkeää, koska siten voidaan varmistaa useista viesteistä tai operaatioista koostuvan palvelun oikea käyttäminen. Protokollien avulla voidaan tuottaa hallintavuo (*control flow*). Hallintavuosta nähdään kuka osapuolista on kulloinkin suorittamassa operaatiota ja vastuussa toiminnallisuuden etenemisestä. Viestiprotokolla sisältää myös viestien sisältöjen väliset riippuvuudet. Näitä riippuvuuksia kutsutaan sisältövuoksi (*data flow*). Esimerkki tilanteesta, jossa tarvitaan sisältövuota on ostotilanne. Ostotilanteessa ostajan antamat osoite- ja laskutustiedot pitää olla mukana ostajalle taikaisin tulevassa laskussa. Sisältövuo on tärkeä, jotta usean osapuolen muodostama kokonaisuus toimisi oikein. Äskeisen esimerkin tapauksessa, jos lisänä on vielä kuljetuspalvelu, joka kuljettaa ostetut tavarat asiakkaalle, tilausnumeron pitää säilyä oikeana sekä asiakkaalle että kuljetuspalvelulle lähetettävissä viesteissä.

Java™-kielessä merkkijono (String) ei ole primitiivityyppi vaan olio.



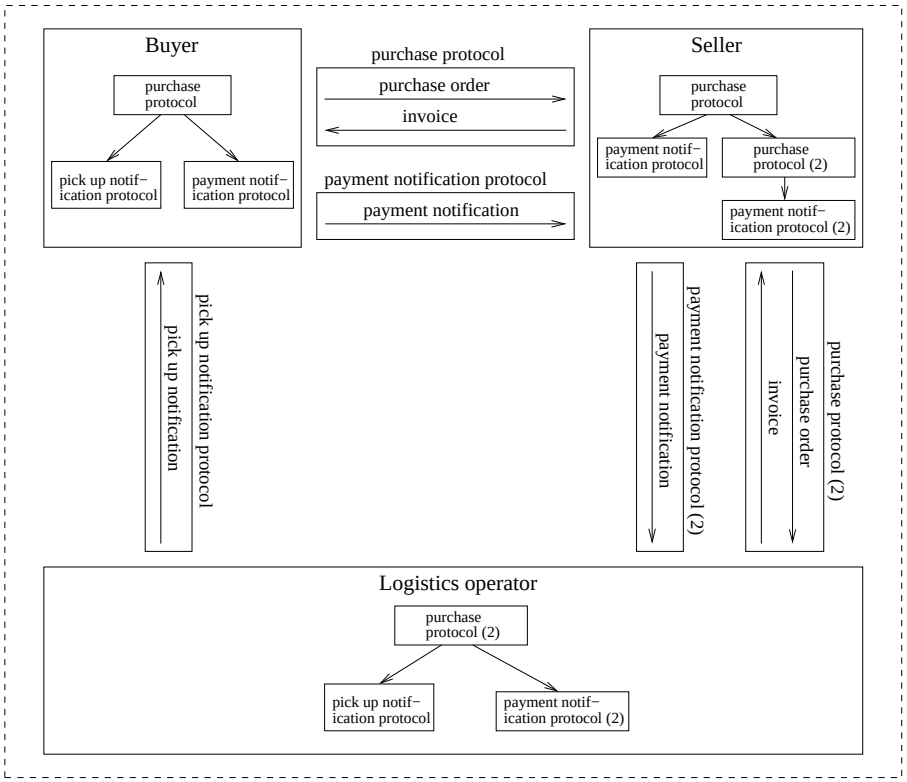
Kuva 3.3: Yksinkertainen työnkulkukuvaus.

Kuvassa 3.3 on kuvattu yksinkertainen työnkulku. Työnkulussa on kaksi osallistujaa *asiakas* ja *myyjä*. Kuvaus kuvaa ostotapahtumaa, jossa asiakas ostaa tuotteita myyjältä. Aluksi asiakas lähettää myyjälle tilauksen ja myyjä vastaa tilauksen käsiteltyään laskulla ostajalle. Myöhemmin asiakas tekee kyselyn tekemänsä tilauksen valmistumisasteesta ja myyjä vastaa kyselyyn. Työnkulkuun liittyvien viestien sisältö on kuvattu työnkulun oikealla puolella. Viestien välille on merkitty katkoviivoin niiden väliset sisältöriippuvuudet.

Toimintakuvauksien ei tarvitse käsittää koko palvelun toimintaa. Kuvauksiin riittää pelkkä ulkoinen kommunikointi muiden osapuolten kanssa. Esimerkiksi kuvassa 3.3 ei esitetä mitenkään sitä kuinka myyjän organisaatiossa koko myyntitapahtuma tehdään. Asiakas näkee muutenkin vain saamansa ja lähettämänsä viestit, joten tieto siitä mitä kaikkia vaiheita myyjän puolella ostotapahtumaan liittyy on jokseenkin turhaa. Tarkemman kuvauksen voi tietysti antaa, mutta sisäisen toiminnan muuttuessa on helpompaa, kun ei tarvitse välttämättä päivittää pelkkää ulkoisen toiminnan kuvausta ollenkaan. Ulkoisen toiminnan esittämiseen on olemassa useitakin erilaisia työnkulunkuvauskieliä kuten WS-BPEL [52] tai WS-CDL [23]. Pelkistettyjä ulkoisia kuvauksia voidaan esittää BPEL:n abstraktien prosessien avulla. BPEL:n abstrakteista prosesseista on poistettu joitakin operaatioita kuvausten yksinkertaistamiseksi eikä niissä ole palveluiden teknisiä sidoksia.

Kuvassa 3.4 on esitetty ulkoiset kuvaukset kohdan 2.2 esimerkkitilanteestaa. Jokaisen osapuolen ulkoiset kuvaukset on jaettu pienempiin osiin, osaprotokolliin. Protokollat nimeävät viestien järjestyksen ja niiden tyypit. Lisäksi jokaisen osapuolen kohdalla on esitetty miten osaprotokollat riippuvat toisistaan, jotta kokonaisuus toimii oikein.

Toimintakuvauksiin voidaan liittää kompensaatiomenettelyjä. Näitä käytetään tapahtumien perumiseen. Peruminen ei kuitenkaan tapahdu peruuttamalla toimin-



Kuva 3.4: Ulkoiset prosessit esimerkkitilanteesta [15].

nallisuutta taaksepäin vaan korvaamalla jo suoritettu toiminnallisuus jollakin tavalla, esimerkiksi rahallisesti. Kompensaatiota voidaan käyttää esimerkiksi tilanteessa, jossa asiakas on tilannut uuden auton maahantuojalta. Asiakkaan taloudellisen tilanteen muuttumisen vuoksi asiakas ei enää pystykään maksamaan uutta autoa, mutta valmistaja on ehtinyt jo valmistaa auton ja maahantuoja kuljettanut auton Suomeen. Peruuttaessaan tilauksen asiakas kompensoi maahantuojalle peruutuksesta aiheutuneita taloudellisia menetyksiä maksamalla ennaltasovitun peruutusmaksun.

Toimintakuvauksiin voidaan vielä lisätä semanttisia kuvauksia viestien ja tarjottujen palveluiden tarkoituksesta. Niiden avulla kerrotaan yksittäisten viestien ja niiden kenttien käyttötarkoitus, jotta kahden palvelun yhteisymmärrys viestin sisällöstä voidaan todeta. Jos palvelut eivät ole samaa mieltä jonkin viestin tai viestin kentän merkityksestä, niiden välille tarvitaan sovitin muokkaamaan viestiä palveluiden välillä tai palvelut eivät ole lainkaan yhteentoimivia. Semanttiset kuvaukset mahdollistavat tarkemman yhteistoiminnan analyysin.

3.4 Kanavamääritykset

Kommunikointikanavat (*channel*) takaavat lopullisen sopimuksen osapuolten tietojärjestelmien välisen matalan tason teknisen yhteensopivuuden. Kanavat kätkevät

sisäänsä kaiken tietoliikenteeseen liittyvän toiminnallisuuden. Kanavan avulla määritellään käytettävä tietoliikenneprotokolla (esimerkiksi TCP/IP) ja siihen liittyvät muut ominaisuudet. Lisäksi kanavan perusteella tiedetään, tarvitaanko kahden tietojärjestelmän välille sovitin (*adapter*) vai ei.

Tietoliikenneprotokollan lisäksi kanavan avulla voidaan määritellä muitakin ominaisuuksia tietojärjestelmien väliselle tietoliikenteelle. Transaktiot ja monenväliset kommunikointiyhteydet ovat tällaisia piirteitä. Kun kanavan tukee transaktioita, se takaa halutun toiminnallisuuden atomisuuden ja mahdollistaa tietynlaisista virhetilanteista toipumisen transaktioiden perumisen (*rollback*) avulla. Transaktioiden lisääminen tietoliikenteeseen pitää kuitenkin ottaa huomioon palveluiden käyttäytymisessä ja kuvauksiin tuleekin merkitä transaktioihin kuuluvat osat käyttäytymistä. Palveluiden tarpeista riippuen sopimukseen tulee valita sopivat kanavat.

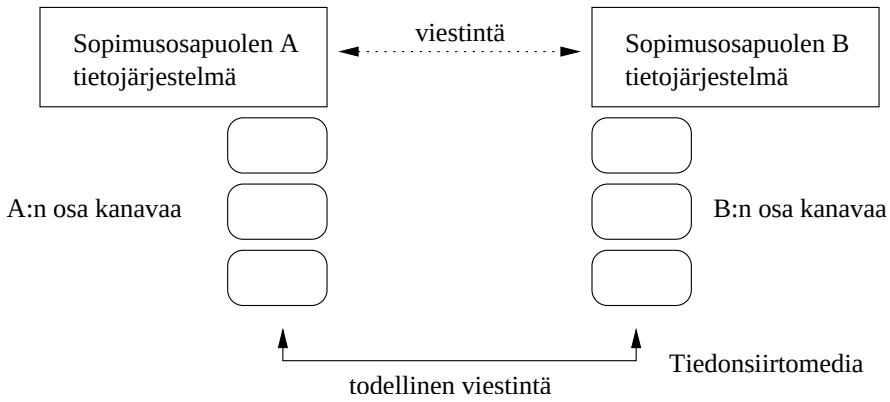
Koska kyseessä on liiketoiminta yritysten välillä, tietoliikenteen tulee olla salattua tarvittaessa. Tietoliikenteen salaamiseen on useita erilaisia vaihtoehtoja kuten julkisen avaimen menetelmät. Jos liikennettä salataan, on sopimukseen liitettävä myös kaikkien osapuolien avaimet. Tietoturvariskien takia on parasta liittää sopimukseen julkiset avaimet ja käyttää niitä symmetrisen salausavaimen luomiseen tarpeen vaatiessa.

Kahden palvelun välille tarvittavat sovitteet voidaan myös liittää tietoliikennekanavaan. Sovittimia tarvitaan tilanteissa, joissa kaksi tietojärjestelmää ei ymmärrä toisiaan ja esimerkiksi tietoliikenne protokolla joudutaan muuntamaan toiseksi ennen kuin liikenne saavuttaa lopullisen palvelun. Sovitinta tarvitaan myös tilanteissa, joissa kahden palvelun korkean tason protokollat eivät aivan toimi yhteen pienien erojen takia. Esimerkiksi tilanteessa, jossa myyjän palvelu tarvitsee vain yhden viestin tietyn transaktion suorittamiseksi ja ostajan sovellus lähettää kaksi viestiä, nämä viestit voidaan yhdistää sovittimessa yhdeksi viestiksi, joka lähetetään myyjän palvelulle.

Kanavilla voidaan myös eristää osapuolten palvelut lopullisista kommunikointiyhteyspisteistä (*communication endpoint*). Tilanteessa, jossa palvelun toteutus siirtyy toiseen osoitteeseen riittää, että kanava konfiguroidaan uudestaan väliohjelmiston sisällä, eikä palvelulle tarvitse tehdä mitään.

Kuvassa 3.5 on esitelty ODP-viitemallin [2] mukainen kanava. Kuvassa on kahden sopimusosapuolen (A ja B) tietojärjestelmät, jotka keskustelevat keskenään jonkin yhteistoimintakuvauksen mukaisesti. Keskustelu kuitenkin tapahtuu tiedonsiirtomedian avulla, joka voi olla vaikkapa kupari- tai valokaapeli. Kanavan muodostaa tietojärjestelmän osan, joka muuntaa sovellukselta lähtevän datan viestiksi kuljetuskerrokselle ja jälleen muuntaa viestin takaisin dataksi sovellusta varten. Muunnoksen yhteydessä käytetään sovittimia ja muita muuntimia tarpeen mukaan.

Luvun 2.2 esimerkkitilanteessa kaikki organisaatioiden väliset kanavat web-palveluosista. Pohjalla on TCP/IP kuljetuskerros. Kuljetuskerroksen päällä käytetään HTTP-protokollaa [12] ja sen päällä vielä SOAP-protokollaa [14]. Nämä muodostavat yhdessä web-palvelu-protokollapinon. Lisäksi osapuolet käyttävät näiden päällä BPEL-ympäristöä, mutta se ei enää varsinaisesti kuulu kanavaan. Toteutuksen yksinkertaistamiseksi esimerkkitilanteen tietoliikennettä ei ole salattu.



Kuva 3.5: ODP-viitemallin mukainen kanava.

3.5 Palvelun laatu

Palvelun laadusta sopiminen on tärkeää tilanteissa, joissa tapahtumien pitää tapahtua tietyissä aikarajoissa ja tietyillä laadullisilla kriteereillä. Tietokoneiden ja hajautettujen järjestelmien palvelun laatu määritellään useimmiten aikarajoilla ja käytettävyydellä (*availability*). Palveluiden välille tarvitsee määritellä muitakin palvelun laadun kriteereitä sovellusten tarpeen mukaan. Sähköisten palveluiden käytössä esimerkiksi käytössä olevan levytilan määrä voi olla merkittävä kriteeri palvelun laadun kannalta.

Palvelun laadun tekniset parametrit voidaan lisätä sopimuksiin. Riippuen parametrin tyypistä ne voidaan sisällyttää suoraan toimintakuvauksiin. Esimerkiksi aikarajat ovat tämäntyyppisiä parametreja. Toiset parametrit, kuten resurssien tarve, voidaan ilmaista sopimuksessa erikseen. Parametreja voidaan myös valvoa automaattisesti tietokoneella, tosin vain rajallisesti [24]. Alkeellisia palvelun laadun parametrejä kuten suoritusaikaa voidaan valvoa.

Puhtaasti sähköisten palveluiden yhteydessä resurssien varaaminen etukäteen palvelunlaadun varmistamiseksi on hyödyllistä. Etukäteen tehtävien varausten avulla palveluntarjoaja voi tarkemmin ennakoida resurssien käyttöä ja havaita mahdollisia ongelmakohtia. Resurssi voi tässä yhteydessä tarkoittaa mitä tahansa sähköistä resurssia tallennuskapasiteetista tietoliikennenopeuksiin ja suorittimen kapasiteettiin.

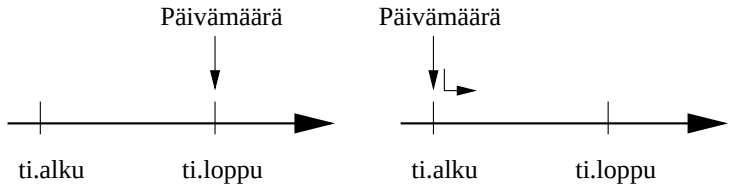
Tehtävä- ja viestipohjaisten järjestelmien palvelunlaatu on samankaltaista konseptien tasolla, mutta toteutustekniikat ovat erilaista [53]. Tehtäväpohjaisella järjestelmä keskittyy tietyn tapahtuman, kuten operaatiokutsun, suorittamiseen ja viestipohjainen järjestelmä keskittyy viestin kuljettamiseen lähettäjältä vastaanottajalle. Tehtäväpohjaisissa järjestelmissä palvelun laatua pyritään takaamaan priorisoinnilla ja tehtävän suorittamisella kriteerien mukaan. Palvelun laadun kontrollointi tehdään olio- tai metoditasolla. Viestipohjaisissa järjestelmissä palvelun laatu pyritään takaamaan viestien vastaanotossa, siirrossa ja prosessoinnissa. Viestipohjaisissa järjestelmissä luotettavuus tarkoittaa muun muassa viestien järjestyksen säilyttämistä

ja uudelleen lähettämistä. Tehtäväpohjaisissa järjestelmissä luotettavuus puolestaan tarkoittaa tehtävän suorittamista loppuun tietyllä todennäköisyydellä.

Web-palveluiden kannalta on olemassa palvelun laadun standardi WS-Reliability [35]. Sen tehtävänä on määritellää luotettavaan viestitykseen liittyviä vaatimuksia. Standardi määrittelee luotettavan viestien vaihdon SOAP-tason protokollalle. Standardi määrittelee luotettavan viestien vaihdon sellaiseksi, joka ei hukkaa viestejä, ei tuota kaksoisviestejä eikä sotke viestien lähetysjärjestystä. Resurssien varaamiseen web-palvelu-ympäristössä ei ole saatavilla standardia tämän opinnäytetyön kirjoittamisen aikana ole. Resurssien käyttämiseen ja niihin viittaamiseen web-palveluiden kautta on kuitenkin olemassa WS-Resource-standardi [39].

3.5.1 Aikarajat

Aikarajat ovat usein ainoa tapa taata, että sovitut asiat tapahtuvat kohtuullisessa ajassa. Jos aikarajoja ei ole, voi jokin sopimuksen osapuoli halutessaan viivytellä jonkin osatoiminnallisuuden suorittamista loputtomiin väittäen etteivät he vielä ole ehtineet sitä tehdä. Kun aikarajat on merkitty sopimukseen, voidaan selkeästi määrittää milloin osapuoli ei ole kyennyt lupaamaansa tehtävää suorittamaan ja siten syyllistynyt sopimusrikkomukseen. Aikarajoja on kahta erilaista tyyppiä. Nämä ovat kova (*hard deadline*) ja pehmeä (*soft deadline*) aikaraja. Kova aikaraja tarkoittaa sitä, ettei sitä saa missään tapauksessa rikkoa. Jos se rikotaan, niin järjestelmän toiminta vaarantuu. Pehmeä aikaraja on enemmän suosituksenomainen ja siitä myöhästyminen ei vaaranna järjestelmän toimintaa. Aikaraja on aina kuitenkin tarkoitettu noudatettavaksi ja sen rikkominen kertoo järjestelmän toiminnassa olevista ongelmista.

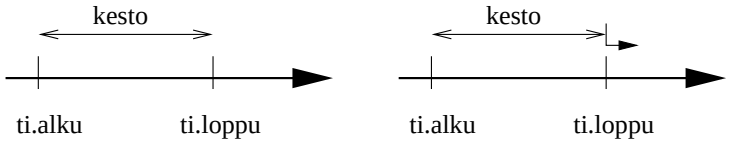


Kuva 3.6: Kova ja pehmeä tapahtuman absoluuttinen aikaraja [30].

Kuvassa 3.6 vasemmalla puolella on esitetty tapahtuman i (ti) päättymiselle kova takaraja ja oikealla puolella tapahtuman i alkamiselle pehmeä takaraja. Kuvassa oikealla puolella olevassa tapauksessa tapahtuman i pitää päättyä aikarajaansa mennessä. Jos se ei pääty aikarajaansa mennessä, seurauksena on sopimuksen rikkominen. Kuvassa oikealla puolella olevassa tapauksessa tapahtuman i tulisi parhaimmassa tapauksessa alkaa aikarajaansa mennessä, mutta se ei haittaa, kunhan tapahtuma käynnistyy mahdollisimman pian aikarajan jälkeen. Absoluuttiset aikarajat on kirjattu esimerkiksi päivämääränä.

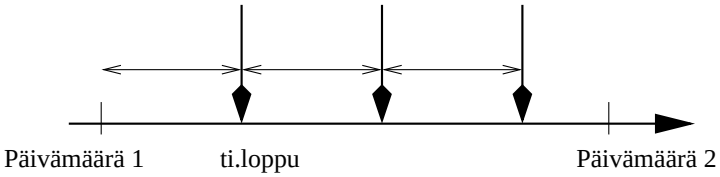
Suhteellinen alkuaikaraja toimii samankaltaisesti, kun edellä mainittu kova aikaraja. Tahtuman i alkuajankohta riippuu toisesta tapahtumasta k. Alkuajankohta

voi riippua tehtävän k alkamisesta tai päättymisestä. Tapahtuma i ei kuitenkaan saa alkaa ennen tapahtumaa k. Jos tehtävä i ei riipu toisesta tehtävästä, tapahtuman suoritusajankohta ei ole merkittävä ja onnistuminen riippuu vain suorituksen kestosta. Tehtävän suorituksen aikarajat voivat liukua yleisen ajan suhteen, mutta päättymisaika on kiinteä suhteessa alkuaikaan. Suhteellisen alkuajan lisäksi tehtävälle voidaan määrittää kova tai pehmeä takaraja. Kova takaraja tarkoittaa kiinteää enimmäissuoritusaikaa.



Kuva 3.7: Kova ja pehmeä tapahtuman keston rajoite [30].

Kuvassa 3.7 vasemmalla puolella on esitetty kova rajoite tapahtuman kestolle ja oikealla puolella pehmeä rajoite tapahtuman kestolle. Tapahtuman kestolle määritelty aikaraja käyttäytyy kuten päättymiselle määritelty suhteellinen aikaraja. Suhteellinen aikaraja voi kuitenkin alkaa aikaisemminkin kuin tapahtuma alkaa. Sen sijaan tapahtuman kestolle määritelty aikaraja alkaa aina silloin, kun tapahtuma alkaa. Kuvassa vasemmalla puolella olevassa tapauksessa tapahtuman kestolle asetetun kovan aikarajan ylittäminen aiheuttaa suoraan sopimusrikkomuksen. Oikeanpuoleisessa tapauksessa kestolle asetetun aikarajan rikkominen ei aiheuta suoraan sopimusrikkoa, koska se on pehmeä.



Kuva 3.8: Toisteisen tehtävän aikarajat [30].

Kuvassa 3.8 on esitetty toisteinen tapahtuma. Toisteinen tapahtuma saa aina uuden takarajan käynnistyessään. Toisteiselle tapahtumalle voidaan asettaa samalla tavalla kova tai pehmeä aikaraja kuten muillekin tapahtumille.

Sopimuksiin kirjattavat aikarajat voivat olla sekä pehmeitä että kovia. Kovia aikarajoja määriteltäessä pitää kuitenkin olla varovainen, koska niiden rikkominen on vakavaa. Monen autonomisen yrityksen toimiessa yhteen kovien aikarajojen noudattamisessa voi olla ongelmia. Tämän vuoksi koviin aikarajoihin pitäisikin jättää tarpeeksi varaa pieniin suoritusaikojen heittoihin.

3.5.2 Resurssien varaaminen

Ajateltaessa pelkästään sähköisiä palveluita kuten vaikkapa tallennuskapasiteetin tarjoamista tai tietoliikennekapasiteetin tarjoamista, pelkästään aikarajoilla ei saada varmistettua palvelun laatua. Tällöin sopimuksessa pitää pystyä ilmaisemaan tarve resurssien varaamiselle. Esimerkiksi juuri tietoliikennekapasiteetin yhteydessä varaaminen voidaan ilmaista tarjottuna siirtokapasiteettina.

Resurssien varaaminen etukäteen on erityisen tärkeää, kun kyse on tosiaikapohjaisesta toiminnasta. Vaikka kaikki liiketoiminta pohjautuu lopulta tosiaikaan, harvoin tarvitaan kovaa tosiaikaisuutta (*hard real-time*) vaan usein pehmeä tosiaikaisuus (*soft real-time*) riittää. Tosiaikajärjestelmiin on kuitenkin kehitelty resurssien varausmenetelmiä, joita voidaan hyödyntää. Yksi esimerkki tällaisesta on RT-CORBA (RealTime - CORBA) [41]. RT-CORBA:n avulla voidaan varata laskentaresursseja etukäteen. Laskentaresursseja voidaan varata säiealtaiden, prioriteettien ja skeduloinnin avulla. RT-CORBA:lla voidaan myös vaikuttaa kommunikointiin säätämällä sen parametreja.

RT-CORBA:n palvelun laadun mekanismit edustavat tehtäväpohjaista (*task*) ajattelua [53]. Mekanismit on suunniteltu helpottamaan tehtävän suoritusajan optimointia. Tämä näkyy säädettävistä parametreista. Esimerkiksi säikeiden prioriteetilla voidaan vaikuttaa tietyn tehtävän suoritusaikaan. Web-palvelu-tekniikka on viestipohjaista, joten palvelun laadun mekanismien pitää myös olla viestipohjaisia. Siksi RT-CORBA:n mekanismeja ei voida suoraa soveltaa palvelunlaadun toteutumiseen. Sen sijaan käsitekehystä voidaan käyttää hyödyksi.

3.6 Luottamus, salaus ja todentaminen

Sopimuksen alaisuudessa tapahtuva toiminta on usein arkaluontoista. Arkaluontoisuus tarkoittaa tässä tapauksessa taloudellisesti merkittävää tai muuten salassapidettävää. Sen vuoksi sopimuksen alla käytettävien palveluiden on oltava luotettavia ja käyttäjien todennettavissa. Tietoliikenne tulee salata ja osia sopimuksestakin voidaan pitää salaisina muilta osapuolilta. Tietoliikenteen salaaminen jää kommunikointikanavan vastuulle.

Sopimuksen tietosisältö on arkaluontoista tietoa, joten se tulisi salata jollain tavalla. Tietyt osat sopimuksista voidaan jopa salata siten, että kaikki osapuolet eivät näe kaikkea tietoa sopimuksesta. Tilanteissa, joissa kahden osapuolen välille sovitaan jotain erityistä, joka ei sido millään tavalla muita, voidaan tämä tieto salata muilta. Yksi esimerkki on yrityskohtaiset tunnukset ja salasanat jonkin toisen yrityksen palveluihin. Tämänkaltainen salaus lisää luottamusta, selkeyttää vastuuvelvollisuuksia ja parantaa sopimuksen valvottavuutta. Luottamus paranee, koska yritykset tietävät tunnusten olevan käytössä vain yhdellä osapuolella. Samalla vastuuvelvollisuuden määrittäminen helpottuu, kun tiedetään kuka osapuolista käytti palvelua ja millä tavalla. Valvottavuus myös paranee osapuolten identifioinnin avulla.

Sopimuksen osittainen salaaminen auttaa myös tilanteisiin, joissa eri osallistujaorganisaatioilla on hyvin erilaiset turvallisuuspolitiikat. Osittaisella salauksella voi-

daan varmistaa, ettei tietty informaatio ole muiden kuin sitä tarvitsevien osallistujien käytettävissä. Osittainen salaaminen ei kuitenkaan ratkaise tilannetta, jossa tiedon tarvitsijoiden joukossa on alhaisen turvallisuustason omaava osallistuja.

Osapuolten pääsynvalvontaan sopimuksen sisällä voidaan käyttää niiden roolia virtuaaliorganisaatiossa. Rooliperustainen pääsynvalvonta on helppo tapa varmistaa, että vain tietty käyttäjä pääsee käsiksi tiettyihin erikoispalveluihin. Erikoispalveluita voi olla vaikkapa virtuaaliorganisaation sisäinen auditointi. Tällöin vain auditoijan roolissa oleva organisaatio pääsee käsiksi auditointiin tarvittaviin palveluihin ja informaatioon. Sen vuoksi sopimuksessa olevat käyttäytymiskuvaukset tulee olla roolitettu. Jos kuvauksina käytetään BPEL-kuvauksia, löytyvät roolit jo valmiina. Roolien lisäksi sopimukseen tulee lisätä vielä organisaatioille erilliset tunnisteet, jotta rooliperustainen todentaminen toimii.

Osapuolten luottamusta toisiinsa voidaan lisätä liittämällä sopimukseen eri osapuolten käyttämiä sertifikaatteja. Sertifikaatteja käytetään yritysten prosessien laadun ilmaisemiseen. Sertifioitujen prosessien, esimerkiksi tiedonkäsittelyn turvallisuuden, käyttäminen lisää luottamusta siihen, että yrityksen toimintatapa on riittävä ja siihen voidaan luottaa. Sertifikaatteja ei tarvitse välttämättä liittää sopimukseen sellaisinaan, vaan listata osallistuvan organisaation asiaankuuluvat sertifikaattien nimet sopimuksessa.

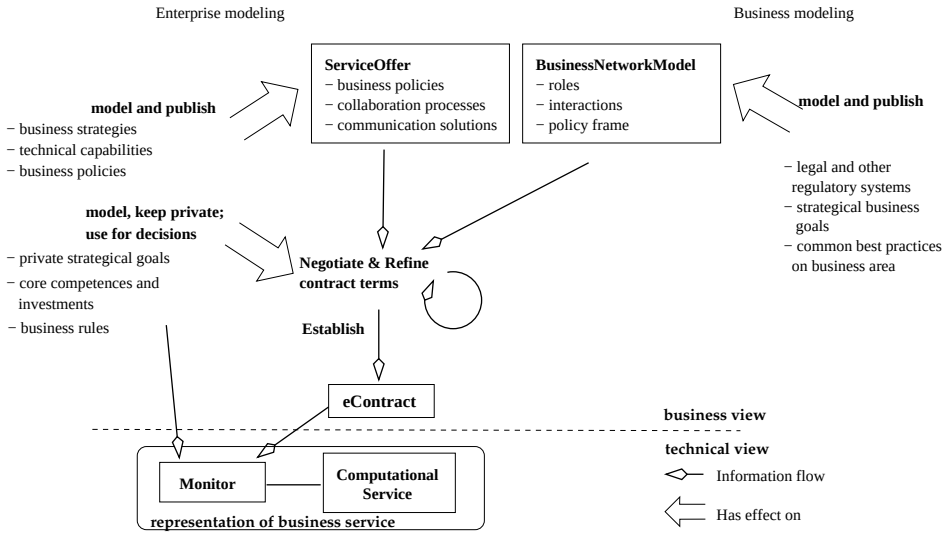
Luku 4

web-Pilarcos sopimustenhallintaympäristö

Tässä luvussa käsitellään web-Pilarcos -projektin prototyypin osaksi toteutettuja sopimusvarastoa ja sopimusoliota. Web-Pilarcos -prototyyppi koostuu useista palveluista, jotka liittyvät sopimusolioon ja sopimusvarastoon. Sopimusolio ja sopimusvarasto ovat osa pienempää alikokonaisuutta jonka nimi on sopimustenhallintaympäristö. Edellä mainittujen komponenttien lisäksi sopimustenhallintajärjestelmään kuuluu verkostonhallinta-agentti. Verkostonhallinta-agentin tehtävä on toimia jäsenen edustajana virtuaaliorganisaatiossa ja suorittaa verkoston hallintaan liittyviä protokollia. Agentti toimii myös sopimusolion apuna organisaatioiden välisessä kommunikoinnissa. Sopimustenhallintaympäristön suunnittelun tavoitteena on ollut luoda aiemmissa osioissa esiteltyjen vaatimusten ja sopimuksen tietosisällön perusteella järjestelmä, joka tarjoaa välineitä tukemaan sopimuksen elinkaarta, sopimuksen tallentamista ja yritysten välistä koordinaatiota sopimuksen elinkaaren aikana.

Sopimuksenhallintaympäristön suunnitteluun vaikuttivat ympärillä olevat komponentit sekä web-Pilarcos -projektin perusolettamukset ympäristön rakenteesta. Projektin peruslähtökohtina ovat nopealiikkeiset globaalit palvelumarkkinat, joissa yritysten tulee pystyä sitoutumaan useihin virtuaaliorganisaatioihin yhtä aikaa. Samalla myös sopimuksen tietosisällön rakenne on vahvasti riippuvainen ympärillä olevista väliohjelmistopalveluista. Tämä näkyy muun muassa siinä, että sopimusten pohjana käytetään verkostomalleja [28], jotka määrittävät sekä virtuaaliorganisaation topologian että jäsenten välille kommunikaatioprotokollat.

Tässä luvussa esitellään asiat seuraavassa järjestyksessä. Aluksi käsitellään toteutusympäristöä ja samalla liitetään varaston ja sopimusolion toteutukset osaksi koko projektissa tuotettua prototyyppiä. Seuraavaksi kerrotaan sopimusvaraston ja sopimusolion toteutusarkkitehtuurista. Lopuksi esitellään varaston ja sopimusolion rajapinnat ja niiden taakse kätkeytyvä toiminnallisuus.



Kuva 4.1: Tiedon siirtyminen suunnittelusta sopimuksiin [28].

4.1 Toteutusympäristö

Web-Pilarcos -projektin lähestymistapa yritystenväliseen yhteistoimintaan on palvelulähtöinen. Samalla mallintaminen perustuu palvelupohjaisiin arkkitehtuureihin (*Service Oriented Architectures, SOA)* [43]. Virtuaaliorganisaatioiden mallintaminen nojaa vahvasti rooliajatteluun ja rooleista koostettaviin kokonaisuuksiin. Mallit verifioidaan etukäteen, jolloin ajonaikainen toiminta on luotettavampaa. Mallit koostuvat metatietokerroksista, jotka rakentuvat toistensa päälle. Alimmalla kerroksella ovat palvelutyypit. Näistä koostetaan rooleja, joista taas koostetaan verkostomalleja [28]. Palveluiden tuottajat julkaisevat palvelutarjouksia, jotka toteuttavat mallinnettuja virtuaaliorganisaation rooleja. Valitsemalla sopivat (eli yhteentoimivat) palvelutarjoukset virtuaaliorganisaation kaikkiin rooleihin saadaan toimiva virtuaaliorganisaatio.

Verkostomalli koostuu yhdestä tai useammasta epookista. Jokaista epookkia kohden on oma arkkitehtuurinsa. Tämä mahdollistaa joustavan verkoston muuttumisen verkoston elinkaaren aikana. Ne kertovat verkoston osapuolten suhteet toisiinsa ja niiden käyttäytymisen verkostossa. Lisäksi verkostomallin määrittelee joukon politiikkoja, jotka ovat olennaisia verkoston toiminnan kannalta. Verkostot koostuvat joukosta rooleja, joihin voidaan valita osallistujia. Roolit on liitetty toisiinsa kommunikaatiokanavin. Roolin muodostaa joukko palvelutyyppejä ja roolikuvaus määrittää kuinka palvelutyypit kokonaisuutena käyttäytyvät. Yksi roolin palvelutyyppi määrittelee keskusteluprotokollan yhtä kommunikaatiokanavaa kohden.

Tiedon siirtymistä sekä liiketoimintamallinnuksesta että yritysmallinnuksesta metatiedon kautta sopimuksiin on kuvattu kuvassa 4.1. Kuvan oikeassa reunassa on liiketoiminnan mallintaminen ja vasemmassa reunassa on yritysten toiminnan mallinnus. Liiketoiminnan mallintamisessa tulee ottaa huomioon lait ja muut säädökset, strategiset liiketoimintatavoitteet ja yleiset toimintatavat tietyllä liiketoiminta-

alueella. Käyttäen pohjalla edellä mainittuja ohjeita mallinnetaan virtuaaliorganisaatiossa tarvittavat palvelut ja niistä koostetaan roolit. Roolien välille mallinnetaan kommunikaatio välitettävine viesteineen, niiden järjestys sekä mahdolliset transaktion ynnä muu sellainen. Lopuksi muodostetaan verkostomallia koskeva politiikka kehikko, joka ohjaa virtuaaliorganisaation toimintaa ja rooleihin valittavia palvelutarjouksia. Tämän jälkeen organisaatiomalli on käytettävissä.

Tuotettavia palveluita mallinnetaan palvelutarjouksilla. Palvelutarjousten osana yritykset kertovat palvelun käyttöön liittyviä politiikkoja, esimerkiksi palvelun hinta. Palvelutarjouksista näkyvät myös mahdolliset palvelun tukemat tekniset kommunikointitavat. Palvelun ulkoinen käyttäytyminen selviää palvelutyypistä. Kaikkeen tähän yritykset käyttävät liiketoimintastrategioitaan ja -politiikkojaan sekä teknistä osaamistaan.

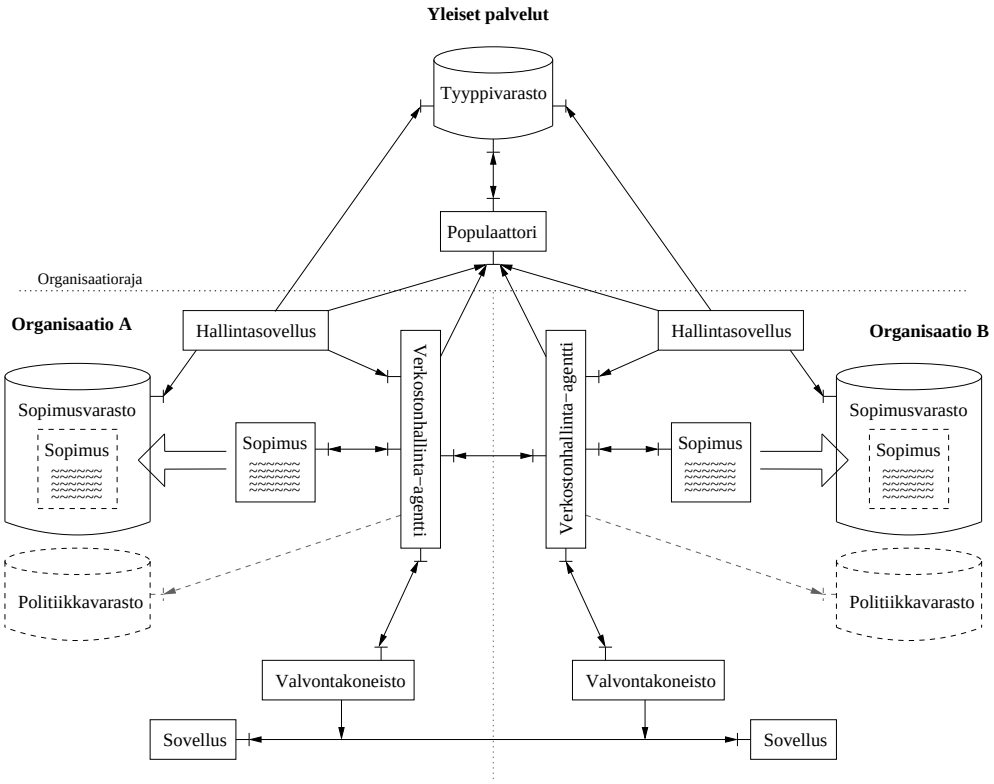
Palvelutarjoukset ja verkostomallit yhdistyvät sopimuksen luomisen yhteydessä. Ennen sopimuksen solmimista osapuolet neuvottelevat sopimuksen ehdoista ja tarkentavat niitä. Neuvottelujen aikana osapuolet pyrkivät varmistamaan mahdollisimman suuren hyödyn omalle toiminnalleen ja käyttävät omia vahvuuksiaan sekä sisäisiä liiketoimintatavoitteitaan. Jos yksimielisyys lopulta saavutetaan, sopimus luodaan ja otetaan käyttöön. Sen jälkeen sopimusta hyödynnetään palvelut tarjoavien sovelluskomponenttien valvontaan.

Esimerkkitoteutuksen ympäristönä on koko web-Pilarcos -prototyyppi. Prototyyppi on suurelta osin toteutettu J2EE-teknologialla [21]. J2EE toteutusalustaksi on valittu JBoss [22], joka on avoimen lähdekoodin J2EE-sovelluspalvelin. J2EE-teknologia mahdollistaa kerrosrakenteisten arkkitehtuurien helpon rakentamisen. Ominaisuutta on hyödynnetty siten, että verkostonhallina-agentti, sopimusolio ja sopimusvarasto muodostavat yhdessä kolmikerroksisen arkkitehtuurin.

Kuvassa 4.2 on esitetty web-Pilarcos -prototyypin palvelut. Kuva jakautuu kolmeen osaan esittäen kahden eri organisaation väliohjelmistopalvelut ja globaalit palvelut. Globaaleja palveluita ovat tyyppivarasto [48] ja populaattori [44]. Jokaisen organisaation on välttämätöntä toteuttaa vain verkostonhallinta-agentin rajapinta, jotta verkostonhallintaan liittyviä protokollia voidaan käyttää. Politiikkavarastoa ei ole prototyypissä toteutettu, mutta se on suunnitteilla ja toteutetaan myöhemmin. Kuvassa olevat isot nuolet kuvaavat sopimusten tallettamista sopimusvarastoon. Sovellusten käyttäytymistä ja viestintää valvotaan erillisen koneiston [16] avulla. Sopimusvarasto, sopimus ja verkostonhallinta-agentti muodostava yhden kokonaisuuden, sopimustenhallintaympäristön. Prototyyppiin kuuluvan hallintasovelluksen on toteuttanut Haba2004 ohjelmistotuotantoprojekti[18]. Sopimustenhallintaympäristöä käsitellään tarkemmin tässä tutkielmassa.

Kuvasta ilmenee web-Pilarcos -mallin hajautettu luonne. Kaikki palvelut lukuun ottamatta tyyppivarastoa ja populaattoria ovat paikallisia. Näin jokaisella organisaatiolla on täydellinen päätäntävalta omien palveluiden toteutustekniikkaan ja sovellusalustaan. Prototyyppi vaatikin vain yhteisen kommunikaatioalustan virtuaaliorganisaation hallintaan. Prototyyppiin valittu hajautus teknologia on web-palvelut[1].

[1]Web-palveluilla tarkoitetaan WSDL:n [8] avulla määriteltyä palvelua, joka käyttää SOAP-protokollaa [14]



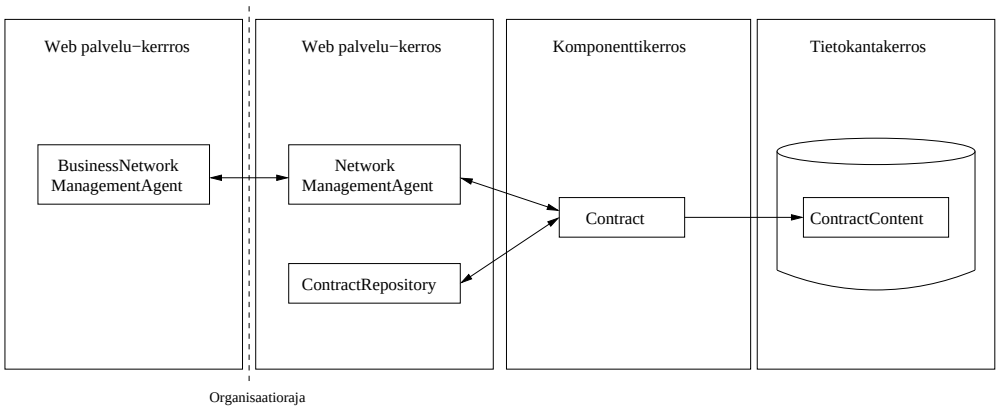
Kuva 4.2: Web-Pilarcos -prototyypin palvelut kahden organisaation kannalta [32].

Vaikka kuvassa sopimusolio ja sopimusvarasto onkin esitetty erillään sopimusoliot tallennetaan sopimusvarastoon ja näin niitä ei voida sijoittaa erilleen toisistaan.

Web-Pilarcos -malli mahdollistaa myös sen, että jollakin osapuolella on vain osatoteutus koko prototyypistä. Esimerkiksi erillistä sopimusvarastoa tai politiikkavarastoa ei välttämättä tarvitse olla toteutettuna mikäli paikallinen järjestelmä pystyy toimimaan verkoston osana ja hallitsemaan omia tietojärjestelmiään ilman kyseisiä komponentteja.

4.2 Toteutusarkkitehtuuri

Sopimustenhallintaympäristö on toteutettu nelikerrosarkkitehtuurina. Komponentit on toteutettu toisistaan eroavasti siten, että sopimusolio on toteutettu J2EE:n *EntityBean*-papuna. Tämän tyyppisellä ratkaisulla saadaan helposti toteutettua tietokannalla tuettu toteutus, joka takaa tiedon pysyvyyden (*persistency*) järjestelmän uudelleenkäynnistysten yli. Verkostonhallinta-agentti ja sopimusvarasto on toteutettu tilattomina *StatelessSessionBean*-papuina. Tämä tarkoittaa sitä, että jokaista kutsua kohden näistä komponenteista luodaan uusi ilmentymä. Tämä helpottaa ratkaisun skaalautuvuutta, koska sovelluspalvelin voi käyttää säiealtaita. Verkostonhallinta-agentit tarvitsevat yhteistä tilatietoa protokollien tilasta, ja so-



Kuva 4.3: Sopimuksenhallintaympäristön kerrokset.

pimusvaraston tarvitsee tietää kaikki organisaation sopimusten tunnisteet. Nämä tiedot löytyvät sovelluspalvelimen tietokannasta.

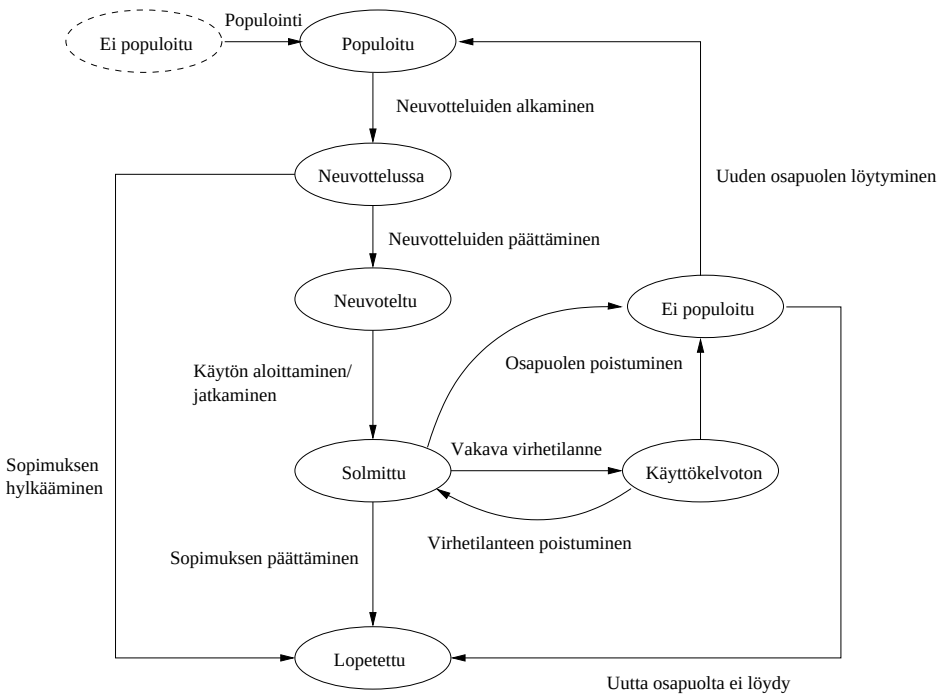
Kuvassa 4.3 on esitetty sopimuksenhallinta järjestelmän jakautuminen kerroksiin. Web-palvelu kerroksella on Verkostonhallinta-agentti (NetworkManagementAgent) ja Sopimusvarasto (ContractRepository). Nämä tarjoavat web-palvelu rajapinnan ulos sopimuksenhallintajärjestelmästä. Verkostonhallinta-agentin rajapinta tarjotaan myös organisaatioiden väliseen kommunikointiin. Sopimusvarasto tarjoaa rajapinnan vain organisaation sisälle. Komponentti kerroksella on itse sopimusolio. Sopimusolioon liittyy sekä sopimusvarasto että verkostonhallinta-agentti. Sopimusolio tallentaa sisältönsä tietokantaan automaattisesti.

Sopimuksenhallintajärjestelmä on suunniteltu siten, että yksi osapuoli sopimuksesta toimii aina koordinaattorina. Koordinaattorin tehtävänä on hoitaa sopimukseen liittyviä yleisiä hallinnointitehtäviä. Tähän kuuluu muun muassa neuvotteluiden edistäminen ja sopimusrikkomusten selvittäminen ja ratkaiseminen.

4.3 Sopimusolio

Sopimusolio on toteutettu mahdollisimman itsenäiseksi päätöksentekijäksi. Kaikkia päätöksiä se ei kuitenkaan tee itse, vaan pyytää päätöstä sopivalta taholta kuten hallintasovellusta käyttävältä ihmiseltä. Esimerkki tämän kaltaisesta tilanteesta on sopimusneuvotteluun osallistuminen, sopimuksesta kieltäytyminen ja sopimuksen hyväksyminen. Sopimus kuitenkin siirtää tilaansa itsenäisesti, eikä siihen voida ulkopuolelta vaikuttaa muuten kuin elinkaarenhallintarajapinnan avulla.

Sopimusolioon toisinnettu siten, että jokaisella organisaatiolla on kopio sopimuksesta. Sopimuksen sisältö siirretään jokaiselle osallistuvalle organisaatiolle neuvotteluiden yhteydessä. Näin jokaisella osapuolella voi olla erilainen toteutus sopimuksesta ja siten tallennusmuodosta. Toteutusten eroista huolimatta sopimuksen tietosisällön tulee olla yhtenäinen kaikkien osallistujien kesken.

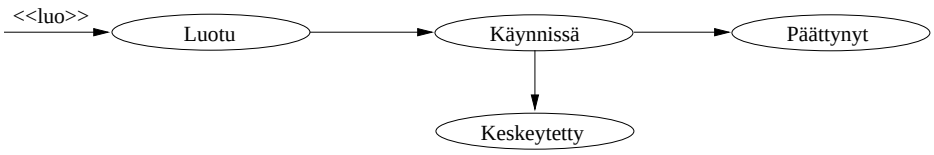


Kuva 4.4: Sopimuksen tilat [32].

Sopimukset luodaan populaattorilta saatavien sopimusehdotusten pohjalta. Halutessaan luoda uuden virtuaaliorganisaation, asiakas pyytää populaattoria etsimään verkostomallin pohjalta yhteentoimivia palvelutarjouksia. Asiakas saa haluamansa määrän tai vähemmän erilaisia sopimusehdotuksia populaattorilta. Määrä voi olla pienempi, koska yhteentoimivia palvelutarjouksia ei välttämättä ole tarpeeksi, jotta riittävä määrä erilaisia ehdotuksia voidaan tuottaa.

Luodut sopimukset vaativat yksiselitteisiä tunnisteita, jotta niitä voidaan käyttää usean organisaation kesken yksiselitteisesti. Sopimusolion tunnisteena käytetään populaattorilta saatujen sopimusehdotusten johdannaisia, koska populaattorin luomien tunnisteiden oletetaan olevan yksiselitteisiä. Siten sopimusten tunnisteetkin ovat yksiselitteisiä. Sopimuksen tunniste on organisaatioiden välinen ja jokainen organisaatio voi halutessaan käyttää sisäistä sopimustunnistetta, kunhan se pystyy muuntamaan organisaation välisen tunnisteen sisäiseksi ja päinvastoin. Organisaation sisäisten tunnisteidenkin on myös oltava yksiselitteisiä.

Kuvassa 4.4 on esitetty sopimusolion tilakaavio ja siihen liittyvät siirtymät. Sopimusolio luodaan populoinnin yhteydessä. Verkostonhallinta-agentti tekee kutsun populaattorille [44] ja muodostaa sopimusoliota populaattorilta saamiensa osallistuja joukkojen perusteella. Tässä yhteydessä sopimuksen tilaksi asetetaan populoitu (*populated*). Ensimmäisen neuvottelukierroksen aluksi sopimuksen tilaksi siirretään neuvottelussa (*innegotiation*). Neuvottelujen päätteeksi sopimuksen tila siirtyy neuvoteltu-tilaan (*negotiated*). Missä tahansa vaiheessa neuvotteluja jokin osapuoli voi hylätä sopimuksen kokonaan, jolloin hylänneen osapuolen kannalta sopimuk-



Kuva 4.5: Sopimusten alaisten istuntojen tilat.

sen tila siirtyy lopetettu-tilaan (*terminated*). Muille osapuolille sopimus siirtyy ei-populoitu-tilaan (*not populated*). Tämä siirtymä ei näy kuvassa.

Neuvottelun jälkeen sopimus solmitaan (*establish*) jonka jälkeen sopimus on käytettävissä. Jos sopimuksen määrittämässä toiminnallisuudessa tapahtuu jokin vakava virhe, sopimus siirtyy käyttökelvoton-tilaan (*unusable*). Tässä tapauksessa ohjelma-agentit eivät osaa yksin ratkaista ongelmaa ja tarvitaan ihmisten osallistumista ongelman ratkaisuun. Kun virhetilanne saadaan selvitettyä, sopimus siirtyy takaisin solmittu-tilaan ja sopimus on jälleen käytettävissä. Sopimuksesta voi jokin osapuoli poistua halutessaan. Tähän syynä voi olla myös toiminnallisuudessa tapahtunut vakava virhe. Kun jokin osapuoli poistuu sopimuksesta, sopimus siirtyy lopetettu-tilaan poistujan näkökulmasta ja muiden näkökulmasta sopimus siirtyy ei-populoitu-tilaan. Tässä vaiheessa sopimuksen jäljelle jääneet osapuolet voivat päättää haluavatko he jatkaa sopimusta etsimällä uuden osapuolen vai ei. Jos uutta osapuolta etsitään ja se löydetään, sopimus neuvotellaan uudelleen uuden osapuolen osalta. Neuvotteluiden jälkeen käyttöä voidaan jatkaa. Uutta osapuolta ei kuitenkaan välttämättä löydetä, jolloin sopimus päättyy ja siirtyy lopetettu-tilaan. Sopimus siirtyy itsestään lopetettu tilaan, kun sen voimassaoloaika joko istuntojen määrän tai päivämäärän perusteella päättyy.

Edellä esitetty kuva sopimuksen tiloista vastaa hyvin luvun 2 osassa 2.4 esitettyä kuvaa sopimuksen elinkaaren vaiheista. Sopimusolion elinkaaressa on enemmän tiloja, ja ne on lisätty kaavioon teknisen toteutuksen helpottamiseksi ja selkeyttämiseksi. Sopimuksen rakentaminen ja neuvottelu rinnastuvat populointi- ja neuvotteluvaiheisiin. Toiminta- ja hallintatila rinnastuu sopimusolion tiloihin solmimisesta eteenpäin päättymiseen asti. Erilaiset virheenhallintatilat kuuluvat hallintaosuuteen.

Kuvassa 4.5 on esitetty sopimusten alaisten istuntojen tilakaavio. Kaavio on suoraviivainen ja virhetilanteiden hallinta on yksinkertaistettu yhdeksi tilaksi (keskeytetty). Suurempaa virhetilanteiden hallintaa ei tarvita, koska sopimuksen tilakaavio tarjoaa virhetilanteiden ratkaisut. Kun sopimus siirtyy pois solmittu-tilasta, kaikki istunnot keskeytyvät automaattisesti.

Sopimusolioon on otettu mukaan kaikkein olennaisin luvun 3 määrittelemästä tietosisällöstä. Tarkka XML skeema sopimuksen tietosisällöstä löytyy liitteestä A. Sopimuksen tietosisältöön kuuluu tieto kaikista sopimuksessa olevista osapuolista. Jokaisesta osapuolesta on joukko tietoa tärkeimpänä sopimuksessa käytettävän palvelun osoite, organisaation hallintarajapinnan osoite ja organisaation allekirjoitus, jota käytetään tunnisteena. Tämä tieto on sopimuksen skeemassa koottu

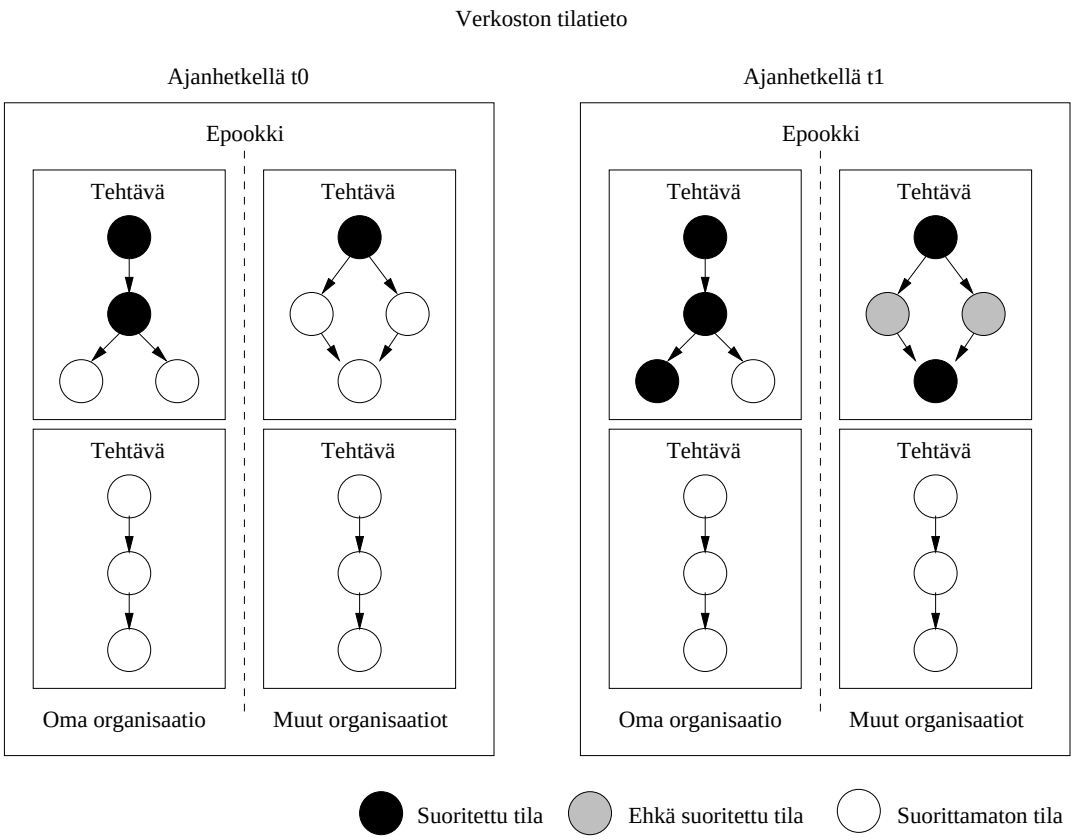
ParticipantInfo-kenttään *ServiceOffer*-tietueessa. Palvelutarjoukset vastaavat kappalessa 3.2 kuvatusta osallistujien informaatiosta. Sopimuksessa on myös sen alku- ja loppupäivämäärä. Loppupäivämäärä voidaan jättää auki sopimuksesta ja sen sijaan käyttää sallittujen istuntojen määrää sopimuksen päättymisen havaitsemiseen. Alku- ja loppupäivämäärät merkataan sopimuksen skeemassa *startDate* ja *endDate*-kenttiin. Sopimus määrittää kuinka monta istuntoa on sallittua suorittaa sen yhteydessä. Sallittujen istuntojen määräksi voidaan myös asettaa ääretön, jolloin istuntojen määrää ei rajoiteta. Istunnot ja niiden määrät ovat sopimuksessa erilaisilla session-loppuisilla kentillä. Alku- ja loppupäivämäärät sekä istuntojen lukumäärä määrittävät sopimuksen voimassaolon.

Verkostomalli on merkitty sopimukseen pelkällä viitteellä ja se haetaan tarpeen mukaan lähimmästä tyyppivarastosta. Toteutuksen sopimuksessa verkostomalli kuvaa palveluiden käyttäytymisen ja palveluiden vaatimuksia. Verkostomallissa oleviin käyttäytymiskuvauksiin on myös liitetty tehtävien suoritukseen asetetut aikarajat. Kuten kerrottu kappaleessa 4.1 verkostomalli määrittelee joukon politiikkoja liittyen malliin. Näille politiikoille on sopimuksessa arvot, jotka on sovittu neuvotteluiden yhteydessä. Sopimus tallentaa erilaisia politiikkoja kolmella tasolla. Nämä ovat verkostomallin taso (*modelPolicies*), yksittäisen arkkitehtuurin taso (*architecturePolicies*) verkostomallin sisällä ja palvelukohtaiset (*rolePolicies*) politiikat. Sopimukseen kuuluu vielä joukko kompensaatioprosesseja, joita voidaan käyttää ongelmatilanteiden ratkaisemiseksi. Näistä prosesseista sovitaan myös neuvotteluiden aikana. Kompensaatioprosessit tulisi kuvata käyttäen sopivaa kieltä kuten BPEL [52] tai WS-CDL [23].

4.4 Istuntojen tilatieto

Istuntoihin liittyvän tilatiedon pohjana on käytetty seuraavia periaatteita. Virtuaaliorganisaation toiminnan tulisi olla synkronoitua siten, että kaikki organisaation jäsenet suorittavat käyttäytymiskuvauksia suunnilleen samalla tahdilla. Näin vältetään ongelmia palveluiden käytön yhteydessä. Synkronointiin käytettävän tiedonmäärän pitää olla mahdollisimman pieni, jotta resurssien tarve on mahdollisimman pieni. Verkostomallin pitää pystyä tukemaan myös evoluutiota. Näistä vaatimuksista kehitettiin tilatietomalli, joka sallii uudelleen konfiguroinnin ja pitää osallistujien välisen kommunikoinnin mahdollisimman alhaisella tasolla.

Kuvassa 4.6 on esitetty istunnoissa käytetty tilatieto. Tilatieto jakautuu kolmeen tasoon, jotka ovat epookit, tehtävät ja koreografiat. Koreografiat koostuvat yksittäisistä viesteistä ja kertovat yksityiskohtaisen ulkoisen käyttäytymisen. Näistä tässä yhteydessä käsitellään vain kahta ylintä tasoa eli epookkeja ja tehtäviä. Koreografioiden seuraaminen kuluu valvontakoneiston [16] vastuulle. Epookit ovat sarjallisia ja ne koostuvat joukosta tehtäviä. Tehtävät puolestaan jakavat koreografiat loogisiin kokonaisuuksiin. Organisaatiot ovat tarkasti tietoisia oman koreografiansa etenemisestä tehtävien sisällä, mutta eivät muiden organisaatioiden etenemisestä. Muiden eteneminen nähdään vain tehtävien tarkkuudella. Tästä syystä kuvan 4.6 ajanhetkessä *t1* organisaatio ei varmasti tiedä kumman tilan jokin toinen organisaatio ko-



Kuva 4.6: Verkoston istuntojen tilatieto.

reografiastaan suoritti, mutta tietää varmasti, että tehtävä on loppuun suoritettu.

Verkostomalli koostuu yhdestä tai useammasta epookista. Epookkien vaihtaminen verkostossa on synkronoitua. Tämä tarkoittaa sitä, että kaikki osapuolet vaihtavat epookkia yhtä aikaa. Tästä on apua tilanteissa, joissa epookin vaihtamiseen liittyy osallistujien lukumäärien vaihtumista tai muuta verkoston uudelleen konfiguroimista. Uudelleen konfiguroituminen voi olla esimerkiksi osallistujan tarjoaman palvelun vaihtuminen toiseen, jolloin paikallisessa tietojärjestelmässä tarvitsee mahdollisesti käynnistää tarvittu palvelu tai vähintään konfiguroida palvelua siten, että verkostoon kuuluvilla muilla organisaatioilla on mahdollisuus käyttää sitä. Lisäksi epookit tarjoavat mahdollisuuden koko organisaation evoluutioon. Tämä voidaan tehdä pysäyttämällä koko organisaatio tiettyyn epookinvaihtoon ja sen jälkeen lisätä organisaatioon uusia osapuolia tai muokata olemassa olevia käyttäytymiskuvauksia jollain muulla tavalla.

Paikallisten palvelukomponenttien hallintaan voidaan käyttää metaohjelmoitavaa väliohjelmistokerrosta [7]. Sen avulla voidaan hallita yksittäisten komponenttien elinkaarta ja niiden uudelleen konfigurointia. Lisäksi kerros pystyy adaptoimaan komponenttien käyttäytymistä. Palvelukomponenttien hallintaan vaadittavista palveluista on tehty projektin puitteissa vaatimusanalyysi [26].

Palautusarvo	Operaatio
void	clear()
boolean	containsContract(java.lang.String contract_id)
boolean	containsSession(java.lang.String session_id)
ContractContent	getContractContent(java.lang.String contract_id)
Contract	getSession(java.lang.String session_id)
boolean	putContract(Contract contract)

Taulukko 4.1: Sopimusvaraston rajapinnan operaatiot.

4.5 Varaston ja sopimuksen rajapinnat

Sopimusvaraston rajapinta on yksinkertainen. Rajapinnan tärkein tehtävä on tarjota mahdollisuus etsiä sopimuksia ja sopimusten alaisia istuntoja. Itse sopimusolion rajapinta tarjoaa pääasiallisesti mahdollisuuksia sopimuksen tietojen manipulointiin. Kuitenkin sopimus suojaa omia tietojaan eli esimerkiksi sopimuksen tilan siirrot edellä kuvatun tilakaavion vastaisesti eivät onnistu.

Sopimusten hallintaan sopimusvarasto tarjoaa kuusi operaatiota. Nämä operaatiot on esitelty taulukossa 4.1. Ensimmäinen operaatio, *clear*, on tarkoitettu sopimusvaraston tyhjentämiseen ja siten kaikkien sopimusten poistamiseen. Yksittäistä sopimustenpoistamisoperaatiota ei ole, koska sopimusten on ajateltu jäävän pysyvästi tietokantaan vaikka niiden voimassaoloaika olisi päättynyt. Tästä on hyötyä, jos tulevaisuudessa sopimuksesta tai sen tuloksista syntyy ristiriitaa. Operaatio *containsContract* on tarkoitettu sopimuksen olemassaolon tarkistamiseen. Tämä operaatio on mahdollisimman kevyt ja sen tarkoitus on toimia nopeana tarkistuskeinona.

Operaatio *containsSession* on suunnattu sopimusten alaisten istuntojen olemassaolon tarkistamiseen. Istuntojen olemassaoloa tarvitaan niiden tilatietojen päivittämisen yhteydessä sopimuksen käytön aikana. Kun halutaan siirtää sopimuksen koko tietosisältö, käytetään *getContractContent*-operaatiota. Tällöin sopimuksen sisältö tilatietoa lukuun ottamatta palautetaan erillisessä tietorakenteessa, joka voidaan helposti siirtää organisaatiosta toiseen. *getSession*-operaatiolla saadaan kahva haluttuun istuntoon sopimuksen alla. Tämä tarvitaan, kun päivitetään istunnon tilatietoa. *putContract*-operaatiolla voidaan tallentaa varastoon uusi sopimus. Tätä käytetään silloin, kun populaattorin ehdotuksista luodaan alustavia sopimuksia ja kun neuvottelun alkuvaiheessa otetaan vastaan sopimusehdotuksia.

Sopimuksen rajapinta on paljon monimutkaisempi. Rajapinnassa on enemmän operaatioita kuin sopimusvaraston rajapinnassa. Koska sopimusolio on toteutettu Java-papuna, rajapinnassa on joukko get- ja set-metodeita, joiden avulla tietosisältöä voi lukea ja muokata. Olioparadigman mukaisesti operaatiot suojaavat sopimuksen tietosisällön eheyttä.

Joukko tärkeimpiä operaatioita sopimusolion rajapinnassa on esitelty taulukossa 4.2. Koordinaattoria varten sopimusolio tarjoaa *changeParticipant*-operaation,

Palautusarvo	Operaatio
boolean	changeParticipant(java.lang.String signature)
int	contractState()
ContractSession	getNewSession(java.lang.String sessionID)
int	getSessionState(java.lang.String session_id)
void	setNewSessionEpoch(java.lang.String sessionID, java.lang.String epochID)
void	terminateContract()
void	updateSessionState(java.lang.String session_id, int new_state)
void	updateState(int new_state)

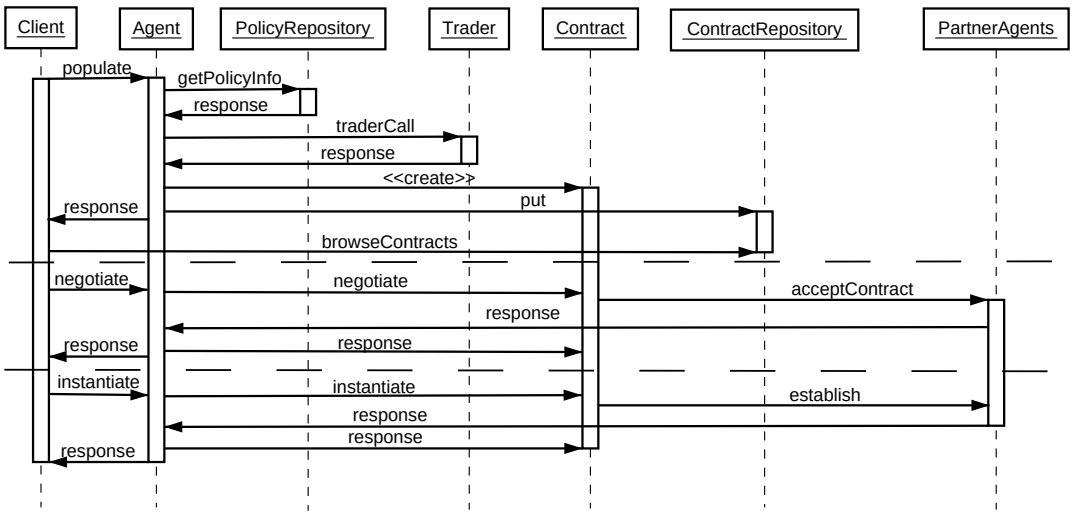
Taulukko 4.2: Sopimuksen rajapinnan operaatiot.

jonka avulla voidaan vaihtaa sopimuksessa olevaa osapuolta. Osallistuja tunnistetaan allekirjoituksen avulla. Operaation tuloksena ei varmasti saada uutta osapuolta, koska markkinoilla ei välttämättä ole sopivaa kandidaattia juuri poistetun osapuolen tilalle. *contractState*-operaatiolla voidaan kysyä sopimusolion tilaa suhteessa aiemmin esiteltyyn sopimusolion tilakaavioon kappaleessa 4.3. Uusia istuntoja sopimukseen voidaan luoda *getNewSession*-operaation avulla. Istunto aloitetaan aina, kun sopimukseen liittyvää toiminnallisuutta aloitetaan käyttämään. Istunnon tilaa voidaan kysyä *getSessionState*-operaatiolla. Istunnon tilaa ei tällä operaatiolla saa tarkasti, vaan tila välitetään karkealla tasolla. *setNewSessionEpoch*-operaatiota käytetään, kun istuntoon liittyvää tilatietoa päivitetään epookin vaihtumisen yhteydessä. *terminateContract*-operaatiolla sopimus päätetään ja samalla se muuttuu muuttumattomaksi. Sopimukseen liittyviä yksityiskohtia ei voida siis enää muuttaa sopimuksen päättymisen jälkeen. *updateSessionState*-operaatio liittyy myös istuntojen tilaan ja sen avulla siirretään istunnon elinkaaren vaiheita eteenpäin järjestelmän itsensä toimesta. Samoin kuin edellisessä operaatiossa *updateState*-operaatiolla päivitetään tilaa, mutta tällä kertaa koko sopimuksen tilaa.

Kaikki edellä mainitut operaatiot sopimusolion rajapinnassa eivät ole tarkoitettu kenen tahansa käyttöön vaan osa on tarkoitettu ainoastaan järjestelmän sisäisille palveluille.

4.6 Sopimustenhallintaprotokollat

Sopimustenhallinta protokollat käsittävät sopimuksen organisaatioiden välisen hallinnan. Näillä protokollilla hallitaan sopimuksen elinkaarta ja virtuaaliorganisaation laajuisten ongelmatilanteiden ratkontaa. Protokollat ovat koordinaattorivetoisia ja levittävät informaation kaikille jäsenille organisaatiossa. Protokollien avulla virtuaaliorganisaation jäsenet pystyvät neuvottelemaan sopimuksen yksityiskohdista ennen sopimuksen pystyttämistä ja äänestämään organisaation jäsenten vaihtamisesta ongelmatilanteiden ratkaisemiseksi. Lisäksi sopimusrikkomuksista voidaan ilmoittaa organisaation koordinaattorille.



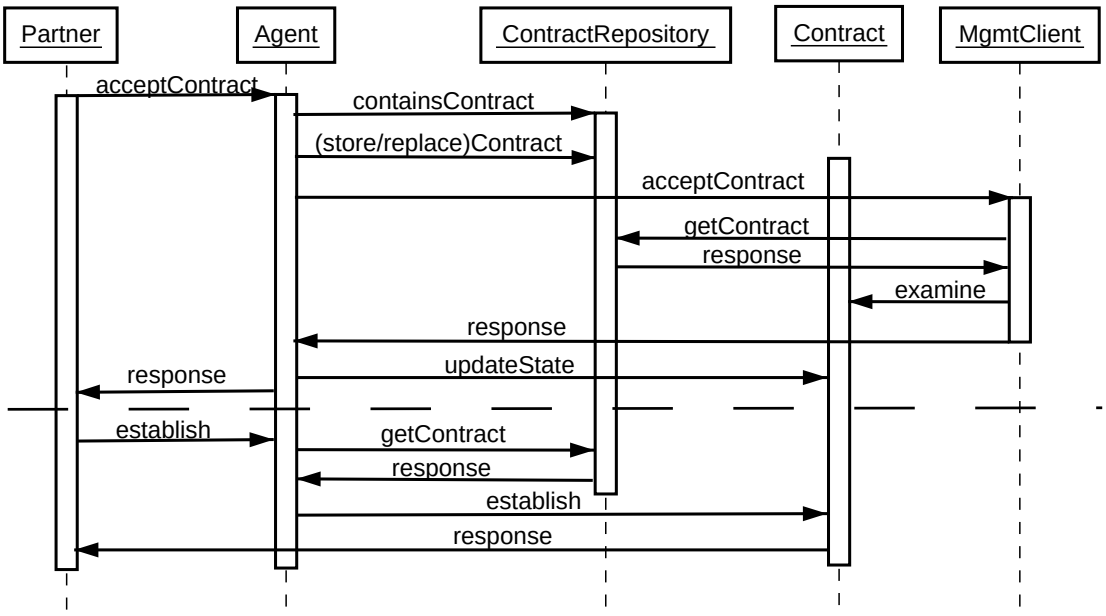
Kuva 4.7: Sopimuksen luominen ja pystyttäminen.

Kuvassa 4.7 on esitetty protokolla verkoston ja sopimuksen luomiseen ja pystyttämiseen. Hallintasovelluksen avulla verkostonhallinta-agentille annetaan käsky populoida jokin tietty verkostomalli. Lisäksi agentille kerrotaan joukko politiikkoja. Agentti hakee nämä politiikat politiikka varastosta ja esitäyttää niiden avulla populaattorille menevän pyynnön. Agentti saa populaattorilta joukon yhteensopivia yhteisöehdotuksia. Agentti luo jokaisesta saamastaan ehdotuksesta sopimusehdotuksen jonka se tallentaa sopimusvarastoon. Lopuksi agentti palauttaa hallintasovellukselle joukon sopimustunnisteita.

Saatuaan joukon sopimustunnisteita hallintasovelluksen käyttäjä voi tarkastella jokaista luotua sopimusehdotusta erikseen. Tarkastelun yhteydessä käyttäjä valitsee sen sopimusehdotuksen, jota hän haluaa käyttää edelleen. Valittuaan haluamansa sopimusehdotuksen hallintasovelluksen käskee hallintajärjestelmää suorittamaan neuvottelukierroksen. Tässä vaiheessa muut sopimukseen valitut organisaatiot saavat ensimmäisen kerran tietää sopimusehdotuksesta. Neuvottelu suoritetaan pyytämällä jokaista sopimuksen osapuolta hyväksymään sopimusehdotus. Jokaisen organisaation toiminta sopimusehdotuksen saadessaan on esitetty myöhemmin kuvassa 4.8. Organisaatiot voivat vastata neuvottelukierrokseen hyväksynnällä, kieltäytymisellä tai vastaehdotuksella. Hallintasovelluksen käyttäjä käynnistää aina uuden neuvottelukierroksen.

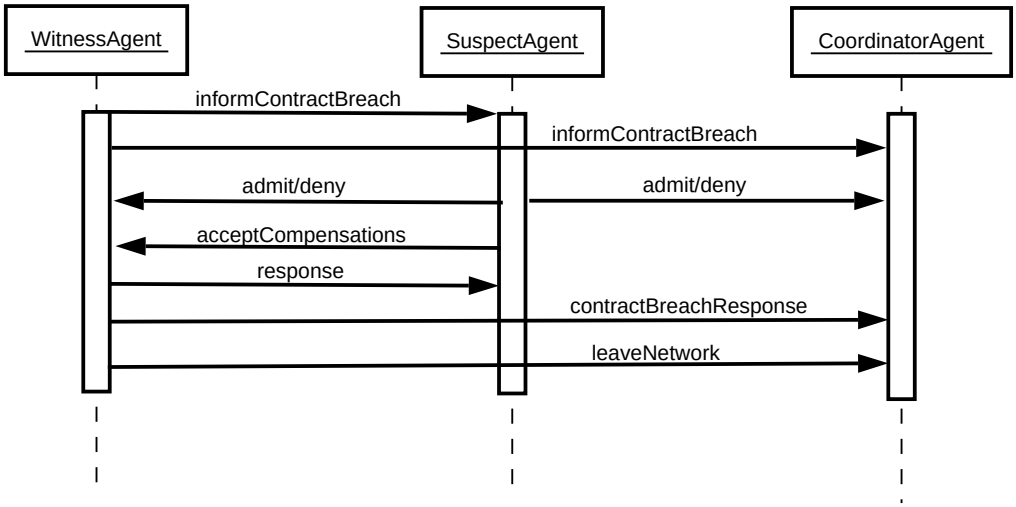
Kun jokainen mukana oleva organisaatio on hyväksynyt neuvotellun sopimuksen, koordinaattori pystyttää sopimuksen pyytämällä kaikkia osapuolia pystyttämään sopimuksen. Sopimusta ei voida pystyttää elleivät kaikki osapuolet ole ilmaisseet hyväksyvänsä sopimuksen. Pystyttämisen seurauksena sopimuksen avulla muodostetun virtuaaliorganisaation tarjoamat palvelut ovat käytettävissä ja sopimuksen määrittelemä toiminnallisuus voidaan käynnistää.

Kuvassa 4.8 on kuvattu organisaation toiminta sen saadessa sopimusehdotuksen hyväksymispyynnön. Saadessaan hyväksymispyynnön agentti tarkistaa aluksi, onko



Kuva 4.8: Pystytys pyyntö.

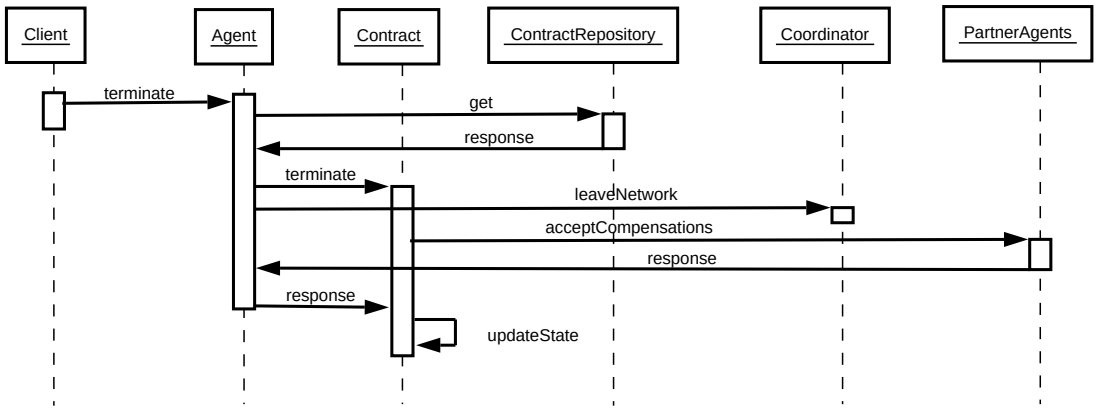
vastaavaa sopimusta jo valmiina sopimusvarastossa vai ei. Jos sopimusta ei löydy, luodaan siitä paikalliseen sopimusvarastoon kopio. Kopion luomisen jälkeen agentti siirtää pyynnön edelleen hallintasovellukselle. Hallintasovelluksen käyttäjä voi tämän jälkeen tarkastella hallintasovelluksen avulla tarjottua sopimusta ja päättää ehdotukseen tehtävistä muutoksista, sopimusehdotuksen suorasta hyväksynnästä tai sopimusehdotuksen hylkäämisestä. Päätöksen jälkeen agentti päivittää sopimuksen tilaa päätöksen perusteella ja vastaa edelleen organisaation ulkopuolelle sopimuksen hyväksynnästä. Vastauksen jälkeen siirrytään taas kuvan 4.7 protokollaan.



Kuva 4.9: Virheiden hallinta.

Kuvassa 4.9 on kuvattu protokolla virhetilanteiden selvittämiseksi. Protokollassa on kolme osapuolta: rikkomuksen havaitsija, rikkomuksen tekijä ja koordinaattori. Kun organisaatio havaitsee sopimusrikkomuksen, se ilmoittaa sekä koordinaattorille että rikkeen tekijälle havainnostaan. Koordinaattorin tehtävä tässä protokollassa on toimia kolmantena osapuolena ja valvoa virhetilanteen ratkaiseminen sovitusti. Rikkeen tekijällä on kaksi mahdollisuutta: joko myöntää rike tai kieltää se. Jos rikkeen tekijä myöntää rikkeen, se suorittaa sopimuksessa sovittut kompensaatiot. Kompensaatioiden suoritus hyväksytetään rikkeen kohteella (useimmiten havaitsija). Rikkeen kohde ilmoittaa myös koordinaattorille, kun se on hyväksynyt kompensaatiot.

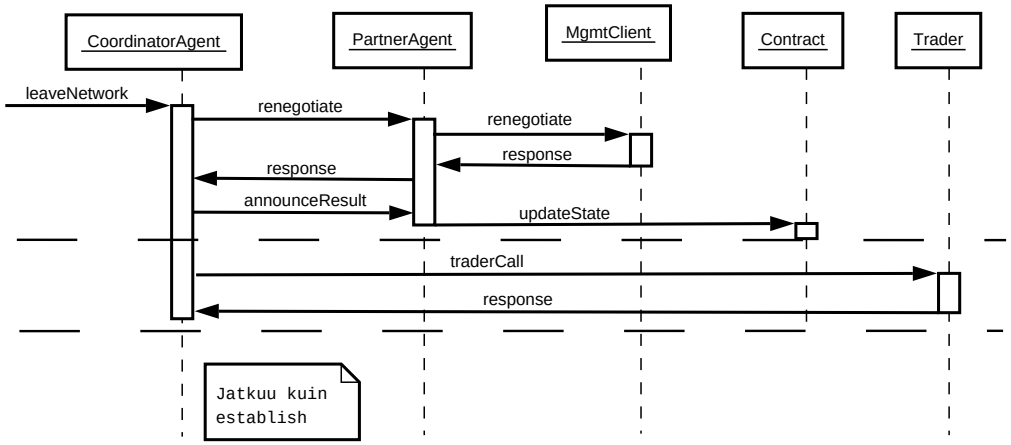
Jos rikkeen tekijä ei tunnusta tehneensä rikettä, rikkeen havaitsijan tai rikkeestä kärsijän tulee päättää onko rike niin suuri että rikkeen tehnyt osapuoli halutaan vaihtaa verkostosta vai ei. Jos rike on tarpeeksi suuri, osapuoli voi pyytää koordinaattorilta rikkeen tehneen osapuolen vaihtoa virtuaaliorganisaatiosta. Tämän jälkeen virtuaaliorganisaation jäljellä olevat jäsenet äänestävät osapuolen vaihtamisesta kuvassa 4.11 esitetyllä tavalla.



Kuva 4.10: Poistuminen.

Halutessaan poistua verkostosta osallistuja antaa *terminate*-käskyn paikalliselle hallintasovellukselle, joka siirtää kutsun eteenpäin agentille kuten kuvasta 4.10 nähdään. Agentti saadessaan kutsun käy hakemassa sopimusvarastosta sopimuksen ja välittää teminate-pyynnön edelleen sopimukselle. Samaan aikaan agentti ilmoittaa koordinaattorille, että organisaatio poistuu sopimuksesta ja siten verkostosta (*leaveNetwork*-viestillä). Koska sopimukseen liittyy todennäköisesti kompensaatioita sopimuksen voimassaolon aikana lähtemiseen, nämä kompensaatiot suoritetaan lähtöilmoituksen jälkeen. Kun kaikki muut osapuolet ovat hyväksyneet kompensaatiot riittäviksi ja suoritetuiksi, sopimus päivittää tilansa terminated-tilaan. Näin sopimus on päättynyt lähtevän osapuolen kannalta. Jos kompensaatioita ei suoritetty hyväksytysti, ongelman ratkaiseminen siirretään hallintajärjestelmän ulkopuolelle.

Uudelleenneuvotteluprotokollaa käytetään, kun jokin osapuoli poistuu tai poistetaan sopimuksesta ja uutta osapuolta tarvitaan lähtijän tilalle. Kuvassa 4.11 on esitetty uudelleen neuvottelu protokolla. Protokolla käynnistyy, kun osapuoli ilmoittaa koordinaattorille lähtevänsä organisaatiosta (*leaveNetwork*-viesti) tai osapuoli



Kuva 4.11: Uudelleenneuvottelu.

pakotetaan lähtemään. Lähdön jälkeen jäljelle jääneet organisaation jäsenet äänestävät uudelleen populoinnista. Koordinaattori lähettää kaikille *renegotiate* viestin jonka osapuolten agentit välittävät omille hallintasovelluksilleen. Hallintasovelluksen avulla käyttäjä vastaa kyllä tai ei, jonka paikalliset agentit taas välittävät takaisin koordinaattori agentille. Koordinaattori kasaa vastaukset ja kertoo vastauksen kaikille osapuolille. Jos verkosto päätettiin uudelleen populoida, niin koordinaattori tekee olemassa olevan sopimuksen perusteella populaattorille kyselyn uusista mahdollisista osapuolista. Koska sopimuksessa on kaikkien osapuolten alkuperäiset tarjoukset, niitä voidaan käyttää uudestaan esitäytettäessä populaattorikyselyä. Sen jälkeen, kun koordinaattori on saanut vastauksen populaattorilta, protokolla jatkuu kuten verkoston pystytysprotokolla kuvassa 4.7.

Luku 5

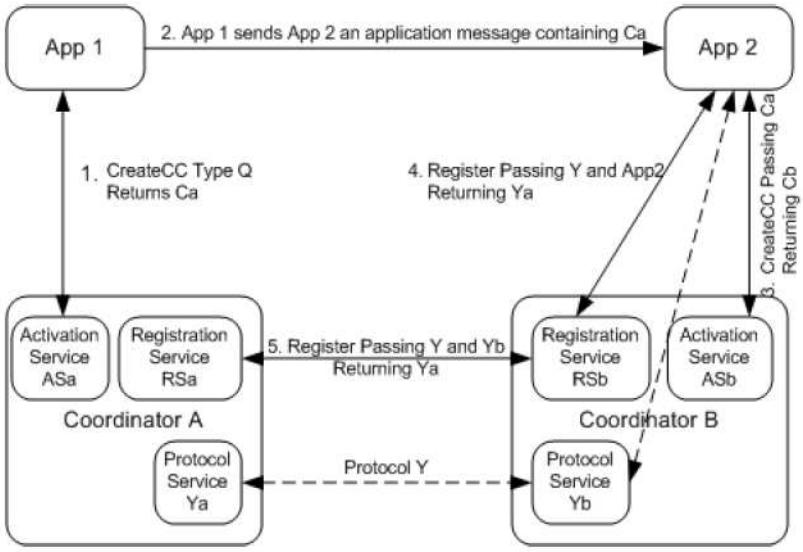
Analyysi

Sopimuksille ja niiden hallintaympäristölle määriteltiin joukko vaatimuksia. Vaatimusten perusteella sopimukselle määriteltiin tietosisältö ja toteutettiin sopimuspohja sekä sopimusten hallintaympäristö. Sopimuksen tietosisältöön vaikuttaa web-Pilarcos -projektin palveluorientoitunut lähestymistapa ja yhteistoiminnan varmentaminen verifioidun metatiedon avulla. Tästä johtuen toteutettu tietosisältö ja luvussa 3 määritelty tietosisältö eivät vastaa täysin tosiaan, mutta kuitenkin sisältävät samat asiat. Sopimuksen ja sen hallintaympäristön toteutus vastaa pääosin määriteltyjä vaatimuksia, mutta parannettavaa jää ja jatkotutkimuskohteita tuli esille.

Tämässä luvussa käsitellään aluksi toteutusta ja verrataan sen ominaisuuksia luvussa 2 esitettyyn tarvekartoitukseen ja toisaalta luvussa 3 esitettyyn sopimusten tietosisältöön. Toteutuksen yhteydessä verrataan prototyyppiin toteutetun globaalin tilan ylläpitoa WS-Coordination-malliin. Molemmat mallit tarjoavat kahden sovelluksen välisen viestienvaihdon synkronointipalvelua. Lopuksi pohditaan kahta isompaa jatkokehittelykohdetta. Nämä ovat neuvottelun toteuttaminen ja virtuaaliorganisaatiossa tapahtuvien muutosten hallinta.

5.1 Vaatimukset ja toteutus

Sopimuspohja täyttää minimivaatimukset sisältönsä suhteen. Vaatimukset olivat virtuaaliorganisaation rakenteen määrittäminen, käyttäytymisen ja siihen liittyvien viestien kuvaaminen, sekä alhaisen tason kommunikaatioprotokollan määrittäminen osapuolten välille. Osapuolten käyttäytyminen ja organisaation rakenne ilmaistaan verkostomallien avulla, joita toteutuksessa ei liitetä suoraan sopimukseen vaan niihin viitataan. Verkostomalli on helposti saatavissa tyyppivarastosta jolloin siitä saatavaa tietoa voidaan käyttää esimerkiksi monitoreiden konfigurointiin. Myös kommunikointikanavat ilmaistaan sopimuksessa viitteellä. Tutkielman perustana olevaan toteutukseen ei ole mallinnettu itse kanavia. Niiden mallintaminen jää toteutettavaksi jälkikäteen. Osallistujien informaatio tallennetaan sopimukseen populaattorilta saatavia palvelutarjouksia apuna käyttäen. Palvelun laatua koskevia määreitä, salausta, autentikointia ja luottamusnäkökohtia ei otettu huomioon toteutuksessa.



Kuva 5.1: WS-Coordinationin arkkitehtuuri [20].

Palvelun laadun kriteerit, varsinkin aikarajat, voidaan kuitenkin mallintaa käyttäytymisen yhteyteen. Osasyynä muiden palvelun laadun määreiden huomiottajättämiselle on se, ettei web-palvelu-ympäristöön ole sopivaa standardia. Todentamisen ja luottamuksen lisäämiseksi prototyyppiin voidaan helpohkosti integroida digitaaliset allekirjoitukset käyttäen olemassa olevaa WSS4J[1]-ohjelmistokomponenttia.

Hallintaympäristö toteuttaa sopimuksen elinkaaren hallinnan. Toteutusteknisistä yksityiskohdista johtuen prototyypin elinkaaren tilakone on monimutkaisempi kuin luvun 2 vaatimuksissa on esitetty. Kaikkia elinkaaren vaiheisiin liittyviä prosesseja ei kuitenkaan ole toteutettu loppuun asti. Näitä ovat muun muassa neuvottelu ja organisaation rakenteen muuttumisen hallinta. Neuvottelua ja muutoksen hallintaa pohditaan lisää kahdessa seuraavassa aliluvussa. Sopimukseen toimeenpanoon liittyvä valvonta on toteutettu erillisessä tutkielmassa [16].

Prototyyppiin toteutettu globaalin tilan ylläpitoprotokolla muistuttaa varsin paljon web-palveluympäristön sovellusten koordinaatio-mallia [20]. Kuvassa 5.1 on esitetty WS-Coordination -mallin arkkitehtuuri ja toiminta kahden sovelluksen viestityksen aikana. Kuvassa on esitetty kaksi osapuolta tai sovellusta ja molempien oma koordinaattori. Aluksi sovellus 1 luo koordinaation kontekstin omalla koordinaattorillaan (A). Tämän jälkeen sovellus 1 lähettää sovellukselle 2 viestin, jonka sisällä on äsken luotu konteksti. Saatuaan viestin sovellus 2 luo oman kontekstin ja antaa luonnissa sovellukselta 1 lähetetyn kontekstin parametrina omalle koordinaattorilleen (B). Kontekstin luonnin jälkeen sovellus 2 rekisteröi saaneensa viestin sovellukselta 1. Tämän jälkeen koordinaattori B rekisteröi koordinaattorille A viestin vastaanoton. Näin molempien sovellusten ulkoisen kuvauksen tila synkronoidaan ja pidetään yhtenäisenä suhteessa toisiinsa.

[1]Web Services Security for Java, http://ws.apache.org/wss4j/

Tutkielman yhteydessä toteutetussa prototyypissä sovellusten ei tarvitse luoda kontekstia itse vaan sen tarjoaa sopimus. Sovellusten ei myöskään tarvitse ilmoittaa saamistaan tai lähettämistään viesteistä vaan monitorit pitävät näistä kirjaa. Lisäksi tilaa ei web-Pilarcos -prototyypissä päivitetä jokaisen viestin jälkeen vaan tehtävien valmistumisen yhteydessä. Tehtävien avulla verkostomallin suunnittelija voi määrittää tilan synkronointivälin haluamakseen. Kuvan koordinaattoreita vastaavat verkostonhallinta-agentit. Ne välittävät muutoksen ulkoisessa tilassa toisilleen.

Web-Pilarcos -mallin ja WS-Coordination -mallin välillä on yksi merkittävä ero. WS-Coordination synkronoi tilaa jokaisen viestin yhteydessä, kun taas web-Pilarcos -prototyyppi päivittää tilaa vain tehtävien tasolla. Web-Pilarcos -mallin globaalin tilan ylläpito on kevyempi suhteessa WS-Coordination -malliin, koska tilan synkronointia tehdään harvemmin. WS-Coordinationin tilan päivitys on kuitenkin tarkempi. Yksittäisten viestien pohjalta tehtävä synkronointi on käytännöllistä vain, jos lähetettäviä viestejä on vähän. Web-Pilarcos -malli ei myöskään vaadi sovelluksilta tietoa koordinointitarpeesta jolloin sovellukset voivat olla yksinkertaisempia.

Sopimuksen laillisuus on jätetty taka-alalle. Toisaalta sopimuksen laillisuus ei ollut alkuperäisissä tavoitteissa. Sopimus onkin rakennettu enemmän teknisistä lähtökohdista. Kuitenkin verkoston mallintaminen siten, että se noudattaa lakeja, tuo lainvoimaisuutta sopimuksellekin. Sopimuksen laillisuuden varmistamiseksi tarvitaan laajaa yritysten välisiä sopimuksia koskevien lakien tuntemusta. Sopimuksen laillisuuden varmistaminen jää kehityskohteeksi.

Vaatimusten mukaan sopimus ja sen hallintaympäristö tulisi olla integroitavissa olemassa oleviin järjestelmiin. Tutkielman osana toteutettu prototyyppi on tehty web-palveluiden avulla. Web-palveluita mainostetaan helposti integroitavaksi tekniikaksi. Muilla tekniikoilla, kuten CORBA tai RMI, toteutetut palvelut voidaan julkaista web-palveluina kääreiden (*wrapper*) avulla. Sopimushallinta on toteutettu itsenäiseksi osajärjestelmäksi, jonka liittymät alla oleviin järjestelmiin ovat lähinnä konfigurointirajapintojen kautta. Tämä puoltaa helppoa integrointia muttei suinkaan takaa sitä. Ainoa tapa todella varmistua järjestelmän integroituvuudesta on kokeilla sitä johonkin olemassa olevaan järjestelmään. Integrointitestiä ei tämän tutkielman puitteissa ole tehty.

Sopimustenhallintaympäristön suorituskykymittauksia ei ole tehty. Hallintaympäristön tulisi kyetä hallitsemaan jopa useita satoja sopimuksia yhtäaikaisesti sekä neuvottelun että muiden elinkaaren vaiheiden osalta, koska yksittäiset organisaatiot voivat olla mukana useissa virtuaaliorganisaatioissa samanaikaisesti. Myös globaalin tilan ylläpitäminen pitäisi onnistua useiden sopimusten osalta samanaikaisesti. Sopimustenhallintaympäristö on pyritty suunnittelemaan skaalautuvaksi. Kerrosarkkitehtuuri tarjoaa useita hajautuspisteitä, joiden avulla yhden tietokoneen kapasiteetin ylittyminen voidaan välttää. Web-palveluiden XML-pohjaisen viestinnän vaikutus on suurempi tekijä. XML-pohjaisen viestinnän ongelmia ovat tekstinä liikutettavat viestit, jotka vievät paljon tilaa, ja mahdollisesti monimutkainen rakenne, jonka tulkinta vie aikaa. Koska komponentit käyttävät keskenäänkin web-palveluviestintää, voi hajautuksella saatava hyöty pienentyä. Hallintaympäristön suorituskyky on myös riippuvainen toteutusalustana käytetyn JBoss-sovelluspalvelimen

suorituskyvystä ja konfiguraatiosta. Hallintaympäristön suorituskykymittaukset on yksi tärkeimpiä jatkotyökohteita.

5.2 Neuvottelu

Hallintaympäristöön toteutettu neuvotteluprotokolla on niin sanottu tinkimisprotokolla (*bargaining*). Erilaisia protokollia on käytännössä kahta tyyppiä [46], huutokauppa (*auction*) ja tinkiminen. Näistä kahdesta päätyypistä on erilaisia variaatioita, esimerkiksi englantilainen huutokauppa ja hollantilainen huutokauppa.

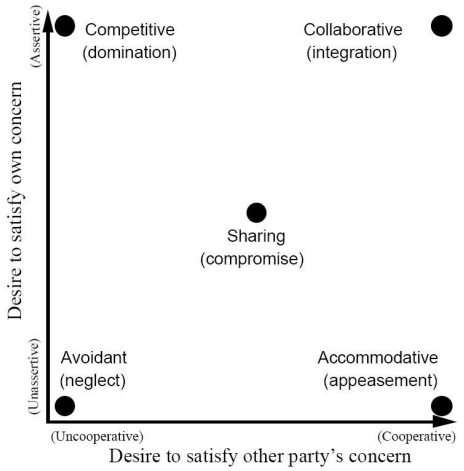
Nykyisessä versiossa neuvotteluprotokolla on kiinteä, mutta se voisi myös olla valittavissa. Erilaisia neuvotteluprotokollia voidaan mallintaa työnkulunkuvauskielillä kuten sopimusosapuolten käyttäytymistäkin. Tulevaisuudessa hallintaympäristöä voisi kehittää siten, että se mahdollistaisi erilaisten neuvotteluprotokollien käyttämisen. Yksi tapa toteuttaa valittavat protokollat on käyttää BPEL-kieltä protokollan kuvaamiseen ja suorittaa protokollaa BPEL-moottorilla. Verkostonhallinta-agentti olisi silloin vastuussa protokollan käynnistämisestä.

Itse päätöksentekoa neuvotteluihin ei ole toteutettu, koska se olisi varsinaisen työkohteen ulkopuolella. Ohjenuorana käyttäytymisen toteuttamiseen voidaan kuitenkin käyttää agenttien käytössä olevia neuvottelujärjestelmiä. Näitä joutuu kuitenkin todennäköisesti muokkaamaan ennen kuin ne sopivat liiketoiminnasta ja sen teknisistä yksityiskohdista neuvotteluun. Perusperiaate on kuitenkin sama.

Neuvottelua voidaan katsoa hajautettuna rajoitteiden (*constraint*) ratkaisuongelmana. Jokaisella osapuolella on omat kiinnostuksen kohteensa ja raja-arvonsa, joiden puitteissa ne suostuvat toimimaan. Näistä vaatimuksista pitää löytää kaikkia tyydyttävä ratkaisu. Paras ratkaisu on tilanne, jossa virtuaaliorganisaatio hyödyttää kaikkia osapuolia mahdollisimman paljon.

Neuvottelun aikana osapuolet voivat käyttää muutamaa erilaista lähestymistapaa [10]. Ne voivat maksimoida omaa hyötyä tai yleistä hyötyä. Yleisen hyödyn huomiointi vaatii tietoa muiden osapuolten tilanteesta ja niille hyödyllisistä asioista. Ongelmana on se, että näitä tietoja ei ehkä haluta paljastaa muille yrityksille. Lisäksi konfliktien hallintaan voidaan käyttää yhteistyöhaluista tai välinpitämätöntä lähestymistapaa. Yhteistyöhaluisuudella tarkoitetaan osallistumista konfliktin ratkaisemiseen ja kompromissien hakua. Välinpitämätön neuvottelija ei ole kiinnostunut konfliktin ratkaisusta ja jättää neuvottelut kesken.

Nämä erilaiset tavat on esitetty kuvassa 5.2. Kuvassa on kaksi akselia, jotka ovat yhteistyöhalukkuus ja huoli omien tavoitteiden täyttymisestä. Esimerkiksi hyvin yhteistyökykyinen neuvottelija, joka ei ole huolissaan omien tavoitteiden täyttymisestä on sopeutuvainen neuvottelija. Kaikkien erilaisten vaihtoehtojen keskellä on kohtuullisen sopeutuvainen ja kohtuullisesti omista tavoitteistaan kiinnipitävä neuvottelija. Olettaen, että kaikki neuvottelijat käyttävät viimeksi mainittua taktiikkaa, neuvottelujen tulos on paras mahdollinen kaikille osapuolille. Tämä ei kuitenkaan ole välttämättä paras taktiikka kaikissa tilanteissa [57]. Yksikin vain omista tavoitteistaan välittävä neuvottelija muuttaa tilannetta ratkaisevasti ja silloin kaikkien muidenkin



Kuva 5.2: Erilaisia käyttäytymismalleja neuvottelussa [10].

kannattaisi välittää vain omista tavoitteistaan.

Erilaisten neuvottelukäyttäytymisten vaikutuksia on jo tutkittu liittyen agentteihin, mutta ei liittyen virtuaaliorganisaatioihin tai liiketoiminnan neuvotteluun. Liiketoiminnassa käytetään lähinnä tietokoneavusteista neuvottelua, jossa ihmiskäyttäjillä on täydellinen päätäntävalta ja tietokone vain antaa suosituksia tarjousten jättämiseen käyttäjän antamien hyötyarvojen perusteella. Agenttipohjaista neuvottelua voitaisiin soveltaa prototyyppiin ja sen neuvottelun päätöksentekoon. Prototyyppi antaa lähtökohdat neuvotteluiden integroimiselle sopimuksen hallintaan. Web-Pilarcos -projektin aikana teetetyn ohjelmistoprojektin [18] tuottamaan hallintakäyttöliittymään voidaan integroida tietokoneavusteisen neuvottelun tarvitsema käyttöliittymä.

5.3 Muutosten hallinta

Virtuaaliorganisaatiossa voi tapahtua muutoksia kahdesta syystä. Joko virtuaaliorganisaation rakenne muuttuu tai organisaation osallistuja vaihtuu tai poistuu. Rakenteen muutoksella tarkoitetaan tilannetta, jossa roolipohjaisesti määritellystä verkostomallista poistetaan tai siihen lisätään rooli suorituksen aikana. Yksittäisen osallistujan vaihtuminen yleensä johtuu väärinkäytöksistä tai muusta räikeästä poikkeumasta sovitussa käyttäytymisessä.

Tilanteita, joissa organisaation rakenne muuttuu ovat esimerkiksi huutokauppatilanteet ja muuttuva logistiikka ketju. Huutokauppatilanteessa organisaatiossa voi olla aluksi useita ostajia ja vain yksi myyjä. Kun sitten huutokauppa on ohi, virtuaaliorganisaatio jatkaa vain kahdella osapuolella, voittaneella ostajalla ja myyjällä. Muuttuvassa logistiikkaketjussa puolestaan voi olla eri määrä toimijoita riippuen kuljetuksen etenemisestä. Kun esimerkiksi kuljetetaan tavaraa maailman toiselle puolelle vesiteitse, kontit kulkevat useiden erilaisten toimijoiden kautta. Näistä

kaikki eivät ole välttämättä ennalta tiedossa.

Molemmat edellä mainitut tapaukset vaativat sopimuksen hallinnalta ja muulta väliohjelmistolta samankaltaisia toimenpiteitä. Aivan ensimmäiseksi väliohjelmiston pitää pystyä arvioimaan muutoksen aiheuttamat vaikutukset käynnissä oleviin transaktioihin. Sen jälkeen virtuaaliorganisaatioon tulee löytää uusi osapuoli ja neuvotella sopimuksen yksityiskohdat uudelleen uuden osapuolen integroimiseksi. Uuden osapuolen löytäminen ja uudelleen neuvottelu on otettu huomioon sopimuksen hallintaprotokollissa. Muutosten aiheuttamien vaikutusten arviointi ei ole mahdollista automaattisesti, mutta prosessin jatkumisen erilaisia vaihtoehtoja voidaan esitellä.

Virtuaaliorganisaation muuttumisen hallinnassa on kuitenkin kaksi haasteellista osaa. Ensinnäkin, miten arvioida osapuolen vaihtamisen vaikutus muihin osapuoliin ja olemassa oleviin transaktioihin, ja toiseksi, miten hallita virtuaaliorganisaation rakenteen muuttuminen silloin, kun käynnissä on useita rinnakkaisia suorituksia, istuntoja, sopimuksen toimintakuvauksista. Kumpaankaan kysymykseen ei ole helppoa ratkaisua.

Jotta osapuolen vaihtamisen vaikutukset voidaan arvioida, tulee organisaation suoritus pysäyttää ja tila synkronoida. Synkronoinnin jälkeen kaikki osapuolet ovat samassa pisteessä ja arvioinnin tulos on yhtenäisempi. Osapuolen vaihtamisen aiheuttamat vaikutukset voidaan päätellä virtuaaliorganisaation pohjalla olevasta verkostomallista. Verkostomalli määrittelee roolien toiminnan ja suhteet toisiin rooleihin. Näistä saadaan erilaiset transaktioketjut eri roolien välillä. Todennäköisesti suurin osa roolien välisistä transaktiosta riippuu jollakin tavalla toisistaan. Ne ovat joko riippuvaisia aikaisemmin suoritettujen transaktioiden tuloksista tai sisäkkäisiä transaktiota. Näin muodostuvat erilaiset ketjut saattavat olla hyvinkin pitkiä. Virtuaaliorganisaation hajautetun luonteen ja erilaisten toteutustekniikoiden vuoksi yhden osallistujan tarjoaman palvelun tilaa ei voida siirtää uuden osallistujan tuottamaan palveluun. Vaihdetusta roolista riippuvat transaktiot täytyy silloin keskeyttää ja aloittaa alusta. Tämä ei kuitenkaan ole mahdollista tilanteissa, joissa materiaalivirtoja liikutetaan paikasta toiseen. Näissä tilanteissa on mahdollisuus käyttää kompensaatioita, joilla voidaan korvata jo loppuun asti suoritettuja osia, joita ei ole mahdollista enää perua.

Käytännössä erilaiset virhetilanteet ja osapuolten vaihtuminen pitää ottaa huomioon jo verkostomallin suunnitteluvaiheessa. Erilaiset virhetilanteet on kuitenkin otettava huomioon ja osapuolen vaihtuminen on lopulta vain yksi virhetilanne lisää. Tosin sen vaikutukset saattavat olla hyvinkin laajat. Olennaisena osana osapuolen vaihtumista on prosessi uuden osapuolen löytämiseksi. Jos verkostomalli tarjoaa erilaisia kompensaatiomenettelyjä tai transaktioiden peruutusmenettelyjä, näistä ja niiden parametreista voidaan neuvotella sopimuksen muodostamisen aikana.

Useat rinnakkaiset istunnot vaikeuttavat muutoksia organisaation osapuolten määrään. Pelkkä vaihtaminen ei vaikuta rinnakkaisiin suorituksiin juurikaan. Sen sijaan osallistujien lukumäärän väheneminen tai lisääntyminen on ongelmallisempaa. Hallittu (ei siis ennalta-arvaamaton) lukumäärän muuttuminen on mahdollista vain epookkien vaihdon yhteydessä. Rinnakkaiset suoritukset voivat olla yhtä aikaa eri epookeissa, mutta yksi suoritus on aina vain yhdessä epookissa kerrallaan.

Osapuolten vähentämisen yhteydessä oikeastaan ainoa järkevä vaihtoehto on osapuolen lopullinen poistuminen sopimuksesta vasta, kun viimeinenkin istunto on ohittanut pisteen, jossa poistuvaa osapuolta ei enää tarvita. Jos osapuoli poistetaan välittömästi, kun ensimmäinen istunto ei enää sitä tarvitse, myöhemmin suoritettaville istunnoille tarvitsee löytää uusi osapuoli tilalle.

Kun osapuolia lisätään epookin vaihtumisen yhteydessä, on kaksi erilaista lähestymistapaa. Ensimmäinen on etsiä aina uusi osapuoli joka kerta, kun istunto saapuu epookin vaihtoon ja tarvitsee uuden osallistujan. Toinen vaihtoehto on valita osallistuja kerran ja käyttää sitä kaikissa istunnoissa.

Ensimmäisen vaihtoehdon huono puoli on se, että aikaa vieviä neuvotteluja joudutaan käymään mahdollisesti useita kertoja riippuen istuntojen määrästä. Lisäksi sopimuksen hallinnan on pidettävä kirjaa jokaisesta osapuolesta ja siitä mihin istuntoon tämä kuuluu. Tämä lisää selkeästi järjestelmän monimutkaisuutta. Toisessa vaihtoehdossa tarvitsee neuvotella vain kerran ja sen jälkeen seuraavat istunnot voivat suoraan käyttää uutta osapuolta. Se ei kuitenkaan tue tilanteita, joissa uuden osapuolen valinta riippuu jonkin transaktion tuloksesta. Jos transaktioiden tulokset eivät ole jokaisen istunnon kohdalla samat, ei samaa osapuolta voida välttämättä käyttää. Toisen vaihtoehdon tuoma joustamattomuus ei tunnu niin pahalta kuin ensimmäisen vaihtoehdon lisäämä monimutkaisuus. Vaihtoehtojen todelliset vaikutukset muun muassa suorituskykyyn tulisi arvioida tulevaisuudessa. Tällä hetkellä toteutus sallii molemmat käyttäytymismallit.

Uuden osapuolen lisääminen tarvitsee joka tapauksessa neuvottelukierroksen. Uuden osapuolen lisääminen epookin vaihdon yhteydessä tulisi vaikuttaa mahdollisimman vähän olemassa oleviin osapuoliin politiikkojen muutoksina ja suoritusalustojen uudelleen konfigurointina. Uuden osapuolen lisäämisen aiheuttama kustannus vaikuttaa suoraan verkostomallin käytettävyyteen ja sen vuoksi lisäämisen kustannukset tulisi selkeästi merkitä verkostomallin kuvaukseen mikäli se on mahdollista. Siten mallin käyttäjät voivat arvioida kustannuksen suuruutta ja päättää haluavatko käyttää mallia vai eivät.

Luku 6

Yhteenveto

Tässä tutkielmassa on tehty tarvekartoitus sopimuksen tietosisällöstä, määritelty sopimuksen tietosisältö, suunniteltu sopimustenhallintaympäristö ja toteutettu hallintaympäristö. Tavoitteena oli myös tutkia olemassaolevien järjestelmien kypsyysastetta. Edustajiksi valittiin ebXML ja OWL-S. Sopimuksen tietosisällön merkittävimpinä ominaisuuksina nähtiin verifioitavuus, valvottavuus ja avoimuus. Verifioinnin avulla voidaan välttää erilaisia ongelmia yhteistoiminnassa kuten esimerkiksi lukkiumia. Valvonnalla puolestaan varmistetaan, että sopimuksen osapuolet pysyvät sopimuksen puitteissa.

Olemassaolevat järjestelmät todettiin puutteellisiksi verrattuna tietosisällön ja hallintaympäristön vaatimuksiin. ebXML:n ongelmina nähtiin kahdenvälisyys ja hallinnan puuttuminen osapuolten löytämistä lukuun ottamatta. OWL-S:n suurin ongelma on puolestaan hallintaympäristön täydellinen puuttuminen. Sopimuksen tietosisällön avoimuutta tuetaan käyttämällä XML-pohjaista sopimuskuvausta jolloin se on käytettävissä kaikilla mahdollisilla alustoilla. Verifioitavuutta ja valvottavuutta tuetaan käyttäytymiskuvauksilla. Kuvausten verifioitavuus riippuu täysin käytetyn kuvauskielen verifioitavuudesta.

Tarvekartoituksen perusteella keskeisiä vaatimuksia sopimukselle ovat sopimuksen laillisuus ja monipuolinen tietosisältö. Tietosisällön keskeisimpiä vaatimuksia ovat virtuaaliorganisaation rakenne, tuki sopimuksen toimeenpanolle, osallistujien vastuualueiden määrittäminen, päivitettävyys ja modulaarisuus. Kartoituksen mukaan pelkkä sopimus ei kuitenkaan riitä, vaan sille on oltava myös tukipalveluita. Tukipalveluilla hallitaan sopimuksen elinkaarta ja tuetaan sen eri vaiheissa tarvittavia erikoispalveluita. Näitä ovat muun muassa osapuolten löytäminen ja neuvottelu.

Sopimuksen tietosisältö määriteltiin koostuvaksi erilaisista teknisistä tunnisteista, käyttäytymiskuvauksista, kommunikaatiokanavista ja palvelun laadun kriteereistä. Käyttäytymiskuvausten pääasiallinen tarkoitus on varmistaa yhteistoiminta mahdollistamalla sekä etukäteinen verifiointi että ajonaikainen sopimuksen toimeenpano. Käyttäytymiskuvausten avulla osallistujat voidaan myös roolittaa ja siten tarjota sopimuksen pohjalta virtuaaliorganisaation rakenne. Kommunikaatiokanavien tehtävänä on varmistaa kommunikointiyhteys kahden osallistujan tietojärjestelmien välille. Yhteyden saaminen ei kuitenkaan takaa yhteistoiminnan onnis-

tumista. Palvelun laadun määreillä tarkennetaan käyttäytymistä ja asetetaan sille ajalliset puitteet. Mahdolliset materiaalivirrat ja niiden kriteerit ovat myös palvelun laadun puolella. Osapuolet tulee saada sidottua sovittuun käyttäytymiseen ja tietovuodot minimoitua. Tämän saavuttamiseksi sopimuksessa on osapuolten sähköiset allekirjoitukset ja osia sopimuksesta voidaan salata.

Hallintaympäristön vaatimusten ja sopimukselle määritellyn tietosisällön pohjalta suunniteltiin ja toteutettiin sopimuksenhallintaympäristö. Hallintaympäristö jakaantuu kolmeen pääkomponenttiin. Nämä ovat itse sopimus(olio), sopimusvarasto ja verkostonhallinta-agentti. Sopimusolioon tallennetaan sopimuksen tietosisältö ja se on vastuussa sopimuksen elinkaaren etenemisestä. Sopimusvaraston tehtävänä on tallentaa sopimukset paikallisesti kuhunkin osallistuvaan yritykseen. Verkostonhallinta-agentin tehtävänä on sekä toimia kontaktipisteenä muille yrityksille että suorittaa verkoston hallintaan liittyviä protokollia. Toteutettujen osien ympärillä on vielä lisätoiminnallisuutta, joka auttaa muun muassa osapuolten löytämisessä sopimuksen muodostusvaiheessa ja sopimuksen valvonnassa.

Hallintaympäristön toteutuksessa ei otettu huomioon kaikkia analyysin vaatimuksia eikä myöskään tietosisällön kokonaisuuksia. Hallintaympäristö kuitenkin toteuttaa elinkaaren hallinnan ja tarjoaa välineet eri vaiheiden suorittamiseen. Ennen kaikkea neuvottelussa ja virhetilanteiden hallinnassa on vielä parantamisen varaa. Sopimuksen tietosisältö on määritelty tyydyttävästi ja se käsittää teknisen yhteistoiminnan varmistamiseen liittyvät seikat. Loppujen lopuksi se kaikkein tärkein seikka eli osapuolten halukkuus yhteistoimintaan havaitaan vasta neuvotteluiden aikana.

Kirjallisuutta

[1] International workshop on Interoperability Solutions to Trust, Security, Policies and QoS. In *Interoperability of Enterprise Software and Applicationsa. Workshops of the I-ESA International Conference 2006.* To appear. `http://istspq2006.cs.helsinki.fi/`.

[2] *Open Distributed Processing Reference Model*, May 2004. `http://www.dstc.edu.au/Research/Projects/ODP/ref_model.html`.

[3] web-Pilarcos-project, May 2006. `http://www.cs.helsinki.fi/group/web-pil`.

[4] Andreoli, J. M., and Castellani, S. Negotiation as a generic component coordination primitive. In *Distributed Applications and Interoperable Systems: 4th IFIP WG6.1 International Conference, DAIS 2003, Paris, France, November 17-21, 2003. Proceedings* (2003), vol. LNCS 2893, Springer Verlag, pp. 86–97.

[5] Bartel, M., Boyer, J., Fox, B., LaMacchia, B., and Simon, E. *XML-Signature Syntax and Processing.* W3C, Feb. 2002. `http://www.w3.org/TR/xmldsig-core/`.

[6] Bidan, C., and Issarny, V. Dealing with multi-policy security in large open distributed systems. In *Computer Security - ESORICS 98: 5th European Symposium on Research in Computer Security, Louvain-la-Neuve, Belgium, September 16-18, 1998. Proceedings* (Sept. 1998), vol. LNCS 1485, Springer Verlag, p. 51.

[7] Breitling, P. Meta-programming middleware for distributed object computing. In *Distributed Applications and Interoperable Systems: 4th IFIP WG6.1 International Conference, DAIS 2003, Paris, France, November 17-21, 2003. Proceedings* (2003), vol. LNCS 2893 / 2003, Springer Verlag, pp. 41–48.

[8] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. *Web Service Definition Language.* World Wide Web Consortium, Mar. 2001. W3C Note (`http://www.w3.org/TR/wsdl`).

[9] Cole, J., and Milosevic, Z. Extending support for contracts in ebXML. In *Proceedings of the workshop on Information technology for virtual enterprises* (2001), IEEE Computer Society, pp. 119–127.

[10] Easterbrook, S. Handling conflict between domain descriptions with computer-supported negotiation. *Knowledge Acquisition 3* (1991), 255–289.

[11] Eisenberg, B., and Nickull, D. *ebXML Technical Architecture Specification.* OASIS, Feb. 2001. `http://www.ebxml.org/specs/ebTA.pdf`.

[12] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. *Hypertext Transfer Protocol – HTTP/1.1.* W3C, 1999. RFC 2616 `ftp://ftp.isi.edu/in-notes/rfc2616.txt`.

[13] Grosof, B. N., and Poon, T. C. SweetDeal: Representing agent contracts with exceptions using XML rules, ontologies, and process descriptions. In *Proceedings of the twelfth international conference on World Wide Web* (May 2003), ACM Press, pp. 340–349.

[14] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., and Nielsen, H. F. *Simple Object Access Protocol.* World Wide Web Consortium, June 2003. W3C Recommendation, `http://www.w3.org/TR/soap/`.

[15] Haataja, J., and Tergujeff, R. Using BPEL4WS for Supply-Chain Integration - Experiences and Evaluation. Tech. Rep. C-2003-74, Department of Computer Science, University of Helsinki and VTT, Dec. 2004.

[16] Haataja, J.-P. Yritysten välisen yhteistoiminnan automatisoitu valvonta. C-2005-17, Department of Computer Science, University of Helsinki, 2005. In Finnish.

[17] He, N., and Milosevic, Z. B2B Contract Implementation Using Windows DNS. In *Proceedings of Workshop on Information Technology for Virtual Enterprises* (2001), IEEE Communications, pp. 71–79.

[18] Hendrikson, R., Kare, A., Lähde, M., Mäki, A., Stenberg, M., and Virtanen, T. HABA2004 ohjelmistotuotantoprojekti, 2004. `http://www.cs.helsinki.fi/group/haba2004/`.

[19] Hofreiter, B., and Huemer, C. B2B Integration - Aligning ebXML and Ontology Approaches. In *EurAsia-ICT 2002: Information and Communication Technology: First EurAsian Conference* (2002), vol. LNCS 2510, Springer Verlag, pp. 339–349.

[20] IBM, Microsoft, BEA et al. *Web Services Coordination*, Aug. 2005. `ftp://www6.software.ibm.com/software/developer/library/WS-Coordination.pdf`.

[21] Java2 Enterprise Edition, May 2006. `http://java.sun.com/j2ee/`.

[22] JBoss, J2EE Application Server, May 2006. `http://www.jboss.org`.

[23] Kavantzas, N., Burdett, D., and et al., G. R. *Web Services Choreography Description language*. W3C, Oct. 2004. http://www.w3.org/TR/2004/WD-ws-cdl-10-20041012/, Working draft.

[24] Klingemann, J. Controlled flexibility in workflow management. In *Proceedings of Advanced Information Systems Engineering: 12th International Conference* (June 2000), vol. 1789/2000, German National Research Center for Information Technology (GMD), Springer-Verlag, p. 126.

[25] Kutvonen, L. *Trading services in open distributed environments*. PhD thesis, Department of Computer Science, University of Helsinki, 1998.

[26] Kutvonen, L. Automated management of inter-organisational applications. In *Eight IEEE International Enterprise Distributed Object Computing* (2002), IEEE Computer Society, pp. 27–38. http://www.cs.helsinki.fi/group/pilarcos/deliverables/kutvonen_management_edoc_2002.pdf.

[27] Kutvonen, L., Metso, J., and Ruokolainen, T. Inter-enterprise collaboration management in dynamic business networks. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE* (Agia Napa, Cyprus, Nov. 2005), vol. 3760 of *Lecture Notes in Computer Science*.

[28] Kutvonen, L., Ruokolainen, T., and Metso, J. Interoperability middleware for federated business services in web-pilarcos. *International Journal of Enterprise Information Systems* (2006). Hyväksytty julkaistavaksi.

[29] Ludwig, H., and Whittingham, K. Virtual enterprise co-ordinator - agreement-driven gateways for cross-organisational workflow management. In *Proceedings of the international joint conference on Work activities coordination and collaboration* (1999), ACM Press, pp. 29–38.

[30] Marjanovic, O., and Milosevic, Z. Towards formal modeling of e-contracts. In *Proceedings of Fifth IEEE International Enterprise Distributed Object Computing Conference* (Sept. 2001), IEEE, pp. 59–68.

[31] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. *OWL-S: Semantic Markup for Web Services*. DAML.org, Feb. 2006. http://www.daml.org/services/owl-s/1.1/overview/.

[32] Metso, J., and Kutvonen, L. Managing virtual organizations with contracts. In *Workshop on Contract Architectures and Languages* (Enschede, The Netherlands, Sept. 2005). http://www.cs.helsinki.fi/group/web-pil/docs/metso_kutvonen_coala2005.pdf.

[33] Milosevic, Z., Berry, A., Bond, A., and Raymond, K. Supporting business contracts in open distributed system. In *Second International Workshop on Services in Distributed and Networked Environments* (1995), IEEE, pp. 60–67.

[34] Neal, S., Cole, J., Linington, P., Milosevic, Z., Gibson, S., and Kulkarni, S. Identifying requirements for business contract language: a monitoring perspective. In *Proceedings of the seventh International Enterprise Distributed Object Computing Conference* (2003), IEEE Communications, pp. 50–61. http://www.cs.kent.ac.uk/pubs/2003/1807.

[35] OASIS. *Web Services Reliability*, Jan. 2003. http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf.

[36] OASIS. *Electronic Business using eXtensible Markup Language (ebXML)*, Feb. 2004. http://www.ebxml.org.

[37] OASIS. LegalXML. Internetissä, May 2006. http://www.legalxml.org.

[38] OASIS. *UDDI Registry - Technical Specification*, Feb. 2006. http://uddi.org/pubs/uddi_v3.htm.

[39] OASIS. *Web Services Resource*, Apr. 2006. http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf.

[40] Open Buying on the Internet, Feb. 2006. http://www.openbuy.org.

[41] Object Management Group. *RealTime - CORBA Specification*, May 2003. http://www.omg.org/cgi-bin/apps/doc?formal/03-11-01.pdf.

[42] Object Management Group. *Common Object Request Broker Architecture*, May 2004. http://www.omg.org/technology/documents/formal/corba_iiop.htm.

[43] Papazoglou, M. J. Extending the service oriented architecture. *Business Integration Journal* (Feb. 2005), 18–21. http://infolab.uvt.nl/pub/papazogloump-2005-40.pdf.

[44] Ponka, I. Populaattori. Tech. rep., Helsingin Yliopisto, http://www.cs.helsinki.fi/group/web-pil/internal/docs/PopulatorImplementationDoc.pdf, Oct. 2004. Projektin sisäinen raportti.

[45] Quirchmayr, G., Milosevic, Z., Tagg, R., Cole, J. B., and Kulkarni, S. Establishment of virtual enterprise contracts. In *Proceedings of the 13th International Conference on Database and Expert Systems Applications* (2002), vol. LNCS 2453, Springer Verlag, pp. 236–248.

[46] Rinderle, S., and Benyoucef, M. Towards the automation of e-negotiation processes based on web services - a modeling approach. In *Web Information Systems Engineering ̈2013 WISE 2005: 6th International Conference on Web Information Systems Engineering, New York, NY, USA, November 20-22, 2005. Proceedings* (2005), vol. LNCS 3806 / 2005, Springer Verlag, pp. 443–453.

[47] Runge, A., Schopp, B., and Stenoevska-Slabeva, K. The management of business transactions through electronic contracts. In *Proceedings of the 10th International Workshop on Database and Expert Systems Applications* (1999), IEEE Computer Society, p. 824.

[48] Ruokolainen, T. Type management functionality in web pilarcos. Tech. rep., Helsingin Yliopisto, Jan. 2005. Projektin sisäinen raportti.

[49] Sollish, F. B. Open Buying on the Internet. Internetissä, Feb. 2006. `http://www.ftc.gov/bc/b2b/comments/obi.pdf`.

[50] Strang, T., Linnhoff-Popien, C., and Frank, K. CoOL: A context ontology language to enable contextual interoperability. In *Distributed Applications and Interoperable Systems: 4th IFIP WG6.1 International Conference* (2003), vol. LNCS 2893, Springer Verlag, pp. 236–247.

[51] Svensson, E., Vetter, C., and Werner, T. Data Consistency in a Heterogeneous IT Landscape: A Service Oriented Architecture Approach. In *Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International* (Sept. 2004), pp. 3–8.

[52] Thatte, S., Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., , Trickovic, I., and Weerawarana, S. *Business Process Execution Language for Web Services*. BEA Systems, IBM, Microsoft, SAP AG, and Siebel Systems. `ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf`.

[53] Wang, G., Chen, A., Wang, C., Fung, C., and Uczekaj, S. Integrated quality of service (QoS) management in service-oriented enterprise architectures. In *Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International* (Sept. 2004), IEEE, pp. 21–32.

[54] Weigand, H., and Hasselbring, W. An extensible business communication language. *International Journal of Cooperative Information Systems 10*, 4 (2001), 423–441.

[55] Weigand, H., and van den Heuvel, W.-J. Cross-organizational workflow integration using contracts. *Decision Support Systems 33* (July 2002), 247–265.

[56] Yellin, D., and Strom, R. Protocol specifications and component adaptors. *ACM Transactions on Programming Languages and Systems 19*, 2 (1997), 292–333.

[57] Zhang, X., Lesser, V., and Wagner, T. Integrative Negotiation Among Agents Situated in Organizations. *IEEE Transactions on Systems, Man, and Cybertetics, Part C: Special Issue on Game-theoretic Analysis & Stochastic Simulation of Negotiation Agents 36*, 1 (2006).

Liite A

Sopimuksen XML Schema

```
<complexType name='ContractContent'>
    <sequence>
        <element name='allowedSessions' type='xsd:int'/>
        <element name='architecturePolicies' nillable='true' type='apachesoap:Map'/>
        <element name='bindings' nillable='true' type='apachesoap:Map'/>
        <element name='businessNetworkModel' nillable='true' type='xsd:string'/>
        <element name='concurrentSessions' type='xsd:int'/>
        <element name='contractID' nillable='true' type='xsd:string'/>
        <element name='conversationRecoveryProcess' nillable='true' type='apachesoap:Map'/>
        <element name='description' nillable='true' type='xsd:string'/>
        <element name='endDate' nillable='true' type='xsd:dateTime'/>
        <element name='globalRecoveryProcesses' nillable='true' type='impl:ArrayOf_Process'/>
        <element name='modelPolicies' nillable='true' type='impl:ArrayOf_tns1_Policy'/>
        <element name='myRole' nillable='true' type='xsd:string'/>
        <element name='participants' nillable='true' type='impl:ArrayOf_ServiceOffer'/>
        <element name='rolePolicies' nillable='true' type='apachesoap:Map'/>
        <element name='roleRecoveryProcesses' nillable='true' type='apachesoap:Map'/>
        <element name='sessions' nillable='true'type='impl:ArrayOf_ContractSession'/>
        <element name='startDate' nillable='true' type='xsd:dateTime'/>
        <element name='state' type='xsd:int'/>
        <element name='usedSessions' type='xsd:int'/>
    </sequence>
</complexType>


<complexType name='ServiceOffer'>
    <sequence>
        <element name='typeID' type='st:TypeID'/>
        <element name='portOffer' type='PortOffer'/>
        <element name='syncStruct' type='st:SyncStruct'/>
        <element name='typingContext' type='st:TypingContext'/>
        <element name='serviceProperty' type='OfferProperty'/>
        <element name='participantInfo' nillable='false' type='ParticipantInfo'/>
    </sequence>
</complexType>


<complexType name='ParticipantInfo'>
    <sequence>
        <element name='address' nillable='true' type='xsd:string'/>
        <element name='contactMail' nillable='true' type='xsd:string'/>
        <element name='contactPerson' nillable='true' type='xsd:string'/>
        <element name='coordinator' type='xsd:boolean'/>
        <element name='location' nillable='true' type='xsd:string'/>
        <element name='managementUrl' nillable='true' type='xsd:string'/>
        <element name='name' nillable='true' type='xsd:string'/>
        <element name='phone' nillable='true' type='xsd:string'/>
        <element name='role' nillable='true' type='xsd:string'/>
        <element name='signature' nillable='true' type='xsd:string'/>
        <element name='url' nillable='true' type='xsd:string'/>
    </sequence>
</complexType>
```

```
<complexType name='ContractSession'>
    <sequence>
        <element name='contractID' type='xsd:string'/>
        <element name='currentEpoch' nillable='true' type='xsd:string'/>
        <element name='sessionID' type='xsd:string'/>
        <element name='state' type='xsd:int'/>
    </sequence>
</complexType>
```