



LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

VANHENTUNEEN SANOMANVÄLITYSOHJELMISTON KORVAUS

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2012
Jouni Virtanen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

VIRTANEN, JOUNI: Vanhentuneen sanomanvälitysohjelmiston korvaus

Ohjelmistotekniikan opinnäytetyö, 48 sivua, 0 liitesivua

Kevät 2012

TIIVISTELMÄ

Asiakkaan käytössä on lankapuhelinverkon laskutusjärjestelmä, joka koostuu eri osajärjestelmistä, ja niissä käytettiin Tuxedo-sanomanvälitysohjelmistoa. Tutkimuksen tavoitteena oli löytää suoraviivainen, yleiskäyttöinen ja kustannustehokas korvaava tekniikka Tuxedo-sanomanvälitysohjelmiston korvaajaksi, koska käytössä olevia Tuxedo-palveluita oli paljon, ja niitä kutsuttiin eri tekniikoilla toteutetuista käyttöliittymistä ja eräajoista. Korvaavan tekniikan täytyi myös olla suorituskyylyltään hyvä. Tämä korvaustyö oli osa isompaa kehitysprojektia.

Tutkimuksessa käytettiin lähteinä Internet-sivustoja ja korvaavan sanomanvälitysohjelmiston manuaaleja. Tutkimusosuusanalyysissä selvisi, että Tuxedoa on käytetty hyvin vähän businesslogiikan toteuttamiseen ja suurin osa palveluista on pelkkää sanomanvälitystä. Lisäksi selvisi, että kaikki kutsut Tuxedo-clienteista on toteutettu C-ohjelmointikielellä. Korvaus tulisi olemaan suoraviivainen sellaisella sanomanvälitysohjelmistolla, joka tukee C tai C++ clientteja ja servereitä.

Korvaavaksi sanomanvälitysohjelmistoksi valittiin TAO CORBA Open Source -ohjelmisto, joka on alun perin tehty Washingtonin yliopistossa. Se on käytössä useissa eri maissa vaativissa sovelluksissa, ja sille saa myös tarvittaessa tukisopimuksen.

Tutkimuksen tavoitteena oli myös löytää ohjelmisto, joka tukee Tuxedo-integraatioiden lisäksi myös muita integraatioita ja jonka suorituskyyky on hyvä. Integraatio-ohjelmistoksi valittiin Oracle WebLogic Server (entinen BEA WebLogic Server). Serveri tukee monipuolisia integraatiotekniikoita esim. Tuxedo ja JMS-sanomajonot.

Korvaavaa sanomanvälitysohjelmistoa testattiin useilla eri testeillä. Kaikki toteutusosuudessa tehdyt testit olivat suoraviivaisia toteuttaa, eikä niissä havaittu ongelmia. TAO CORBA -serverit vastasivat nopeasti clienttien kutsupyntöihin, joten korvaavan sanomanvälitysohjelmiston suorituskyyky todettiin myös hyväksi. TAO CORBA:n kuormantasauksella servereihin saatiin myös lisää skaalautuvuutta ja lisäksi asynkroniset palvelukutsut nopeuttivat clienttien toimintaa.

Korvausprojekti on jo valmistunut ja käytännön kokemukset TAO CORBA -sanomanvälitysohjelmistosta sekä WebLogic Server -integraatiosta ovat olleet hyviä. Ohjelmistojen suorituskyyky on havaittu testeissä erinomaiseksi.

Avainsanat: Tuxedo, TAO CORBA, sanomanvälitysohjelma, WebLogic Server

VIRTANEN, JOUNI: Replacement of obsolete middleware software

Bachelor's Thesis in software engineering, 48 pages, 0 appendix

Spring 2012

ABSTRACT

Our customer is using a fixed line billing system, which contains several subsystems and all these systems were using Tuxedo middleware software. The purpose of this study was to find a straightforward, generic and cost-effective technique to replace the existing Tuxedo middleware software, because our customer had a lot of Tuxedo services, Tuxedo batch job clients and Tuxedo user interface clients implemented with different kinds of techniques. The performance of the replacing technique also had to be good. This replacement work was a part of a bigger development project.

Internet sites and the manuals of the replacement middleware software were used as a source of information in this study. The end result of the analysis was that only a minor part of the billing system was using Tuxedo in its business logic, and the major part of the Tuxedo services were basically message delivering between client and server. Additionally, it became clear in the analysis that all service calls in the Tuxedo clients were implemented by using the C programming language. The replacement of the Tuxedo middleware software would be straightforward with such software that supports C or C++ clients and servers.

TAO CORBA Open Source middleware software was chosen as a replacement for Tuxedo. It has originally been made at the University of Washington and today it is widely used in several countries in demanding applications. It is also possible to get a commercial support contract for it if needed. Additionally, WebLogic Server was chosen as integration software because it supports versatile integrations, for example Tuxedo and JMS queues.

The replacement middleware software was tested in several ways in the implementation part of this study. All these tests were straightforward and there were no issues detected in the tests. TAO CORBA servers rapidly replied to clients' requests, so the performance of the replacement middleware software was also found good. TAO CORBA LoadBalancing features also increased the scalability of the servers and, additionally, asynchronous service calls made the clients' execution faster.

The implementation of the replacement project is ready and hands-on experiences about TAO CORBA middleware software and WebLogic Server integration software have been good. The performance of the software has been found excellent in the tests.

Key words: Tuxedo, TAO CORBA, middleware software, WebLogic Server

SISÄLLYS

1	JOHDANTO	1
2	TOIMINTAYMPÄRISTÖN KUVAUS	3
3	TUTKIMUSONGELMA JA TUTKIMUKSEN TAVOITE	4
3.1	Odotettavat tulokset ja rajaukset	4
3.2	Projektin aikataulu	4
4	TUTKIMUSOSUUS	6
4.1	Kuluttaja-asiakkaiden tilaustietojärjestelmän analysointi (APM)	6
4.2	Yritysasiakkaiden tilaustietojärjestelmän analysointi (Atlas)	6
4.3	Liikennetikettien hinnoittelujärjestelmän analysointi (Lihäs)	6
4.4	Laskutusjärjestelmän analysointi (Lasso)	7
4.5	Kutsumäärät tietyltä ajanjaksolta ulkoisten järjestelmien palveluihin	7
4.6	Kutsumäärät tietyltä ajanjaksolta järjestelmän omiin palveluihin	7
4.7	Tutkimusosuuden yhteenveto	8
5	TOTEUTUSOSUUS	9
5.1	Korvaavan sanomanvälitysohjelmiston valinta	9
5.1.1	Korvaavan sanomanvälitysohjelmiston toimintaperiaate	11
5.1.2	Korvaavan sanomanvälitysohjelmiston testaus	13
5.1.3	Monitasoisen arkkitehtuurin testaus korvaavalla sanomanvälitysohjelmistolla	13
5.1.4	Kuormantasauksen testaus korvaavalla sanomanvälitysohjelmistolla	16
5.1.5	Asynkronisten sanomien testaus korvaavalla sanomanvälitysohjelmistolla	18
5.1.6	IDL-tiedoston kääntäminen lähdekoodiksi	21
5.1.7	Make Project Creator työkaluohjelman käyttäminen	23
5.1.8	TAO CORBA -serverin toimintaperiaate	24
5.1.9	TAO CORBA -clientin toimintaperiaate	26
5.1.10	TAO CORBA -nimipalvelu	27
5.2	Integraatio-ohjelmiston valinta	29
5.2.1	Integraatio-ohjelmiston toimintaperiaate	29
5.2.2	Tuxedo-domainin kuvaus	32

5.2.3	Integraatio-ohjelmiston komponentit	34
5.2.4	Integraatio-ohjelmiston testaus	35
5.2.5	Tuxedo-sanomien reititys	36
5.2.6	Palomuurin aukipitäminen ja ulkoisten Tuxedo-domainien tunnistus	36
5.2.7	JMS XML request / reply -sanomien kohdistus ja poikkeustilanteiden käsittely	37
5.2.8	Stateless session bean Java-luokan toimintaperiaate	37
5.2.9	Integraatiosovelluksen luokkakaavio	37
5.2.10	Integraatiosovelluksen BaseBean-luokan toiminnallisuus	40
5.2.11	Integraatiosovelluksen Ash_tLiitTuot-luokan toiminnallisuus	42
6	YHTEENVETO, JOHTOPÄÄTÖKSET JA KÄYTÄNNÖN KOKEMUKSET	45
6.1	Laskutusjärjestelmän uusi arkkitehtuuri	46
6.2	TAO CORBA -sanomanvälitysohjelmiston tulevaisuus	47
	LÄHTEET	49

TERMIT JA LYHENTEET

APM (OE 2)

Kuluttaja-asiakkaiden tilaustietojärjestelmä.
Order Entry system 2.

Atlas (OE 1)

Yritysasiakkaiden tilaustietojärjestelmä.
Order Entry system 1.

Lihäs (Rating)

Puhelinkeskuksilta tulevien liikennetikettien hinnoittelujärjestelmä.

Lasso (Billing)

Lankapuhelinverkon laskutusjärjestelmä.

SAMI

Vikatikettien ylläpitojärjestelmä.

Alpha

Tilaustietojärjestelmä.

CDM

Kuluttaja-asiakkaiden mastertietojärjestelmä.

CSI

Integraatiojärjestelmä.

IB2

Integraatiojärjestelmä.

SNET

Verkkotietojärjestelmä.

SQL*Net

Structured Query Language Network.

Tuxedo

Transactions for Unix, Extended for Distributed Operations.

CORBA

Common Object Request Broker Architecture.

JMS

Java Message Service.

EJB

Enterprise JavaBeans.

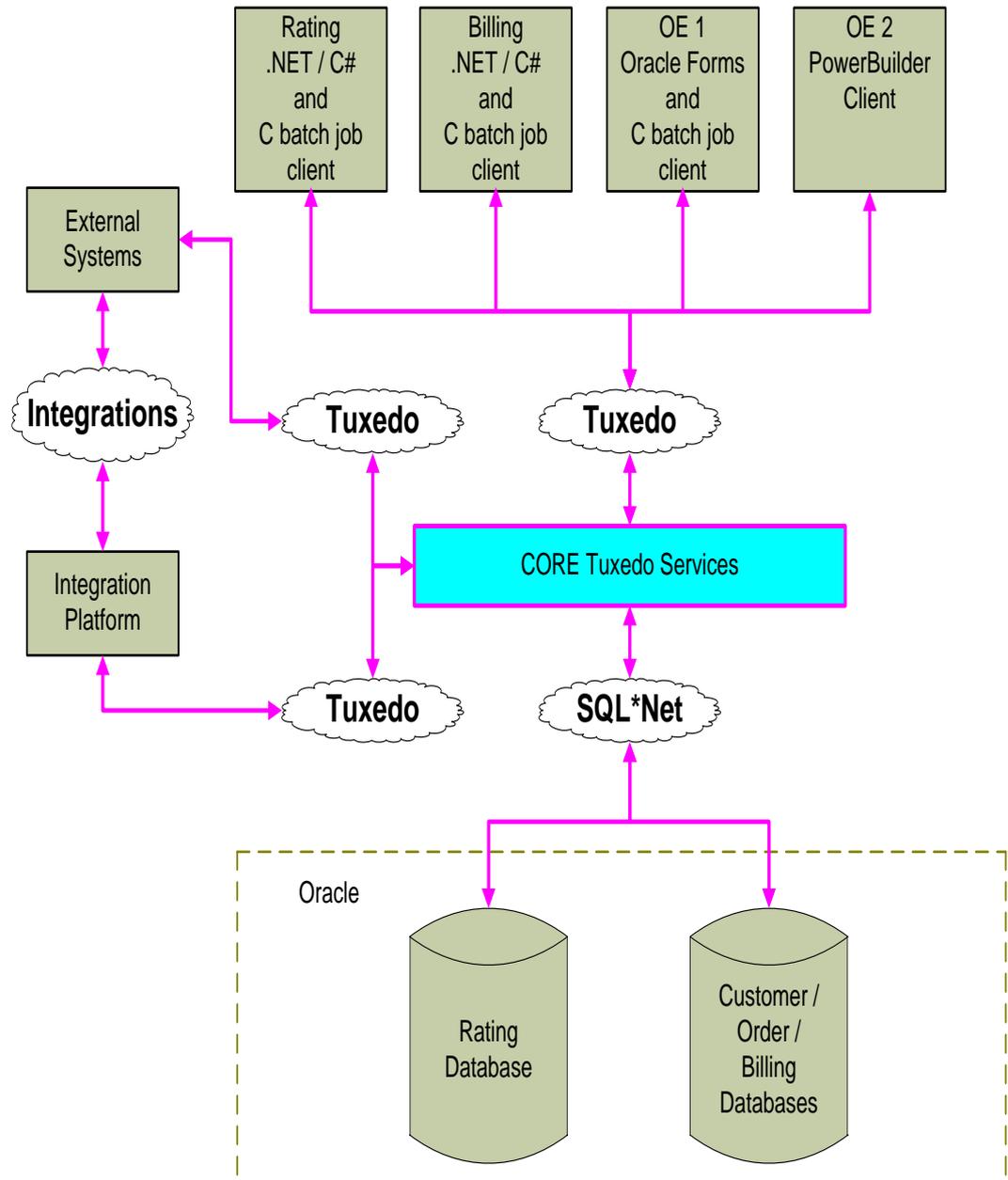
1 JOHDANTO

Tämä opinnäytetyö on tehty TeliaSonera Finland Oy:n broadband alueen IT-laskutusosastolle, ja työn valvojana sekä tilaajana asiakkaan puolella on toiminut IT-laskutusosaston osastopäällikkö. Projekti aloitettiin vuoden 2010 tammikuussa, valmistui vuoden 2012 tammikuussa, ja se vietiin tuotantoon kolmessa eri vaiheessa. Työskentelin opinnäytetyötä tehdessäni ohjelmistoarkkitehtinä Tieto Oy:n ja Iriba Oy:n palveluksessa tämän projektin aikana ja jatkan edelleen työntekijänä Iriba Oy:n palveluksessa integraatioarkkitehtinä.

Asiakkaan käytössä on lankapuhelinverkon laskutusjärjestelmä, joka koostuu seuraavista osajärjestelmistä:

- kuluttaja-asiakkaiden tilaustietojärjestelmä (OE 2 eli APM)
 - muutosten tuotantoon siirto vaiheessa 2
- yritysasiakkaiden tilaustietojärjestelmä (OE 1 eli Atlas)
 - muutosten tuotantoon siirto vaiheessa 1
- puhelinkeskuksilta tulevien liikennetikettien hinnoittelujärjestelmä (Rating eli Lihas)
 - muutosten tuotantoon siirto vaiheessa 1
- laskutusjärjestelmä (Billing eli Lasso)
 - muutosten tuotantoon siirto vaiheessa 1.

Jokaisessa järjestelmässä on useita eri tekniikoilla toteutettuja käyttöliittymiä, eräajoja ja Tuxedo-sanomanvälitysohjelmiston välityksellä käytettäviä palveluita. Osajärjestelmät tarjoavat palveluita myös ulkoisille järjestelmille ja myös käyttävät ulkoisten järjestelmien palveluita. Nykyisin käytössä oleva sanomanvälitysohjelmisto ei ole enää asiakkaan tietotekniikka-arkkitehtuurissa (kuvio 1) hyväksytty tekniikka, ja asiakas haluaa korvata tämän vaihtoehtoisella tekniikalla.



KUVIO 1. Alkuperäinen laskutusjärjestelmän arkkitehtuuri

2 TOIMINTAYMPÄRISTÖN KUVAUS

Tässä luvussa kuvataan kuviossa 1 kuvatun laskutusjärjestelmän tarkempi toiminnallisuus. Kuluttaja-asiakkaiden tilaustietojärjestelmällä (OE 2) on käytössä Windows PowerBuilder -käyttöliittymiä. Yritysasiakkaiden tilaustietojärjestelmällä (OE 2) on käytössä Oracle Forms -käyttöliittymiä, ja järjestelmän eräohjelmat on toteutettu C- ja Pro*C -ohjelmointikielillä. Laskutusjärjestelmällä (Billing) ja liikennetikettien hinnoittelujärjestelmällä (Rating) on käytössä C# .NET -käyttöliittymiä, ja järjestelmien eräohjelmat on toteutettu C- ja Pro*C -ohjelmointikielillä.

Laskutusjärjestelmän ydinpalveluita CORE Tuxedo Services käytetään osajärjestelmien käyttöliittymistä ja eräajoista, ja tiettyihin ydinpalveluihin on tarjottu kutsurajapinta ulkoisille järjestelmille. Ulkoisiin järjestelmiin on toteutettu suoria integraatioita, ja osa integraatioista on toteutettu asiakkaan integraatio-ohjelmiston IB2 kautta. Tietokannan käsittelyyn käytetään Oraclen SQL*Net -protokollaa Structured Query Language Network.

Kuluttaja-asiakkaiden tilaustietojärjestelmällä, yritysasiakkaiden tilaustietojärjestelmällä ja laskutusjärjestelmällä on yhteinen tietokanta (Customer / Order / Billing Databases). Liikennetikettien hinnoittelujärjestelmällä on oma tietokanta (Rating Database).

Tilaustietojärjestelmillä ylläpidetään asiakkaan tilaustietoja, ja liikennetikettien hinnoittelujärjestelmä hinnoittelee asiakkaan puhelut ja laskutusjärjestelmä muodostaa asiakkaan puheluista laskun. Asiakkaan laskulla on kiinteitä kuukausimaksuja, ja lisäksi laskulla voi olla puhelinliittymän käyttöön perustuvia puhelumaksuja. Laskutusjärjestelmä välittää laskutusajossa muodostetun laskutusaineiston ja asiakkaiden asiakastiedot erilliseen kokoavaan laskutusjärjestelmään, joka yhdistelee asiakkaan kaikki laskut yhdeksi laskuksi. Asiakkaalle toimitetaan esimerkiksi lankapuhelinverkon sekä mobiiliverkon laskutus samalla laskulla.

3 TUTKIMUSONGELMA JA TUTKIMUKSEN TAVOITE

Ongelmana on löytää suoraviivainen, yleiskäyttöinen ja kustannustehokas korvaava tekniikka Tuxedo-sanomanvälitysohjelmiston korvaajaksi, koska käytössä olevia Tuxedo-palveluita on paljon, ja niitä kutsutaan eri tekniikoilla toteutetuista käyttöliittymistä ja eräajoista. Korvaavan tekniikan täytyy myös olla suorituskyvyltään hyvä.

Tutkimuksessa käytetään seuraavia menetelmiä eli kyseessä on ns. sekamuoto, joka koostuu eri menetelmistä:

- induktiivinen menetelmä ja aineistolähtöinen menetelmä
- kokeellinen menetelmä ja konstruktiiivinen menetelmä.

3.1 Odotettavat tulokset ja rajaukset

Teknisen ratkaisun täytyy olla asiakkaan IT-arkkitehtuuriin soveltuva ratkaisu, ja sen täytyy myös sisältää ulkoisten järjestelmien integraatorajapinnan toteutus. Toisin sanoen kuviossa 1 kuvattuja ydinpalveluita CORE Tuxedo Services täytyy pystyä käyttämään järjestelmän sisällä, ja myös tiettyihin järjestelmän ydinpalveluihin täytyy tarjota kutsurajapinta ulkoisille järjestelmille. Tässä projektissa korvaavaksi sanomanvälitysohjelmistoksi valittiin TAO CORBA ja integraatio-ohjelmistoksi Oracle WebLogic Server (entinen BEA WebLogic Server).

3.2 Projektin aikataulu

Projekti aloitettiin tammikuussa 2010. Atlas-, Lasso-, ja Lihas-järjestelmien muutokset vietiin tuotantoon tammikuussa 2011. APM-järjestelmän muutokset vietiin tuotantoon kesäkuussa 2011. Integraatio-ohjelmisto vietiin tuotantoon tammikuussa 2012. Projektin aikataulu on esitetty Gantt-kaaviona kuviossa 2.

4 TUTKIMUSOSUUS

Tutkimusosuudessa analysoidaan seuraavat asiat:

- Tuxedo-palveluiden ja palvelimien määrä osajärjestelmittäin
- osajärjestelmien väliset ristiinkutsut palveluiden osalta
- kutsumäärät tietyltä ajanjaksolta ulkoisten järjestelmien palveluihin
- kutsumäärät tietyltä ajanjaksolta järjestelmän omiin palveluihin
- Tuxedon osuus businesslogiikan toteuttamisessa
- Tuxedo-clienttien toteutustekniikka.

4.1 Kuluttaja-asiakkaiden tilaustietojärjestelmän analysointi (APM)

Kuluttaja-asiakkaiden tilaustietojärjestelmällä on käytössä 379 Tuxedo-palvelua ja 37 Tuxedo-palvelinta. Osajärjestelmä käyttää kahta liikennetikettien hinnoittelujärjestelmän palvelua. Tuxedoa ei ole käytetty businesslogiikan toteuttamiseen vaan ainoastaan sanomanvälitykseen. Tuxedo-clientit on toteutettu PowerBuilder-käyttöliittyminä, joissa on myös käytetty Microsoft Visual Studio C DLL -tekniikkaa. Kaikki Tuxedo-kutsut on toteutettu C-ohjelmointikielellä synkronisina kutsuina. Tuxedo-palvelut on toteutettu C- ja Pro*C -ohjelmointikielillä.

4.2 Yritysasiakkaiden tilaustietojärjestelmän analysointi (Atlas)

Yritysasiakkaiden tilaustietojärjestelmällä ei ole omia Tuxedo-palveluita. Osajärjestelmä käyttää neljätoista kuluttaja-asiakkaiden tilaustietojärjestelmän palvelua. Tuxedo-clientit on toteutettu Oracle Forms -käyttöliittyminä ja C-eräajoina. Kaikki Tuxedo-kutsut on toteutettu C-ohjelmointikielellä synkronisina kutsuina. Tuxedo-palvelut on toteutettu C- ja Pro*C -ohjelmointikielillä.

4.3 Liikennetikettien hinnoittelujärjestelmän analysointi (Lihäs)

Liikennetikettien hinnoittelujärjestelmällä on käytössä 65 Tuxedo-palvelua ja 8 Tuxedo-palvelinta. Osajärjestelmä tarjoaa kaksi palvelua kuluttaja-asiakkaiden tilaustietojärjestelmälle. Tuxedoa on käytetty businesslogiikan toteuttamiseen yh-

den palvelun osalta, jota kutsutaan käyttöliittymästä. Tuxedo-clientit on toteutettu C# .NET -käyttöliittyminä ja C-eräajoina. Käyttöliittymissä on myös käytetty Microsoft Visual Studio C DLL -tekniikkaa. Kaikki Tuxedo-kutsut on toteutettu C-ohjelmointikielellä synkronisina ja asynkronisina kutsuina. Tuxedo-palvelut on toteutettu C- ja Pro*C -ohjelmointikielillä.

4.4 Laskutusjärjestelmän analysointi (Lasso)

Laskutusjärjestelmällä on käytössä 12 Tuxedo-palvelua ja 8 Tuxedo-palvelinta. Osajärjestelmä käyttää yhtätoista liikennetikettien hinnoittelujärjestelmän palvelua. Tuxedoa on käytetty businesslogiikan toteuttamiseen yhden palvelun osalta, jota kutsutaan käyttöliittymästä. Tuxedo-clientit on toteutettu C# .NET -käyttöliittyminä ja C-eräajoina. Käyttöliittymissä on myös käytetty Microsoft Visual Studio C DLL -tekniikkaa. Kaikki Tuxedo-kutsut on toteutettu C-ohjelmointikielellä synkronisina ja asynkronisina kutsuina. Tuxedo-palvelut on toteutettu C- ja Pro*C -ohjelmointikielillä.

4.5 Kutsumäärät tietyltä ajanjaksolta ulkoisten järjestelmien palveluihin

Analyysi tehtiin kuluttaja-asiakkaiden tilaustietojärjestelmän kutsumääristä ulkoisten järjestelmien palveluihin ajanjaksolta 10.11.-17.11.2010, ja analyysissä oli mukana 4 ulkoista järjestelmää. Kutsujen kokonaismäärä tältä ajanjaksolta oli 9222 kpl.

4.6 Kutsumäärät tietyltä ajanjaksolta järjestelmän omiin palveluihin

Analyysi tehtiin ulkoisten järjestelmien kutsumääristä kuluttaja-asiakkaiden tilaustietojärjestelmän omiin palveluihin ajanjaksolta 4.9.-4.9.2009, ja siinä oli mukana 21 palvelua. Kutsujen kokonaismäärä tältä ajanjaksolta oli 5355 kpl.

4.7 Tutkimusosuuden yhteenveto

Tuxedoa on käytetty businesslogiikan toteuttamiseen vain kahdessa palvelussa. Tutkimusosuudessa analysoidut määrät on esitetty myös taulukkomuodossa TAULUKKOISSA 1 - 3.

TAULUKKO 1. Tuxedo-palvelujen ja Tuxedo-palvelimien määrät

Osajärjestelmä	Tuxedo-palvelut	Tuxedo-palvelimet
Kuluttaja-asiakkaiden tilaustietojärjestelmä (APM)	379	37
Yritysassiakkaiden tilaustietojärjestelmä (Atlas)	0	0
Liikennetikettien hinnoittelujärjestelmä (Lihäs)	65	8
Laskutusjärjestelmä (Lasso)	12	8
Yhteensä	456	53

TAULUKKO 2. Osajärjestelmien väliset Tuxedo-palvelujen ristiinkutsut

	APM	Atlas	Lihäs	Lasso
APM	0	0	2	0
Atlas	14	0	0	0
Lihäs	0	0	0	0
Lasso	0	0	11	0

TAULUKKO 3. Kuluttaja-asiakkaiden tilaustietojärjestelmän kutsut ulkoisiin järjestelmiin ja kutsut ulkoisista järjestelmistä järjestelmän omiin palveluihin

Ajalta	Ulkoisten järjestelmien määrä	Kutsujen määrä
10.11 – 17.11.2010	4	9222
Ajalta	Palvelujen määrä	Kutsujen määrä
4.9.2009 – 4.9.2009	21	5355

5 TOTEUTUSOSUUS

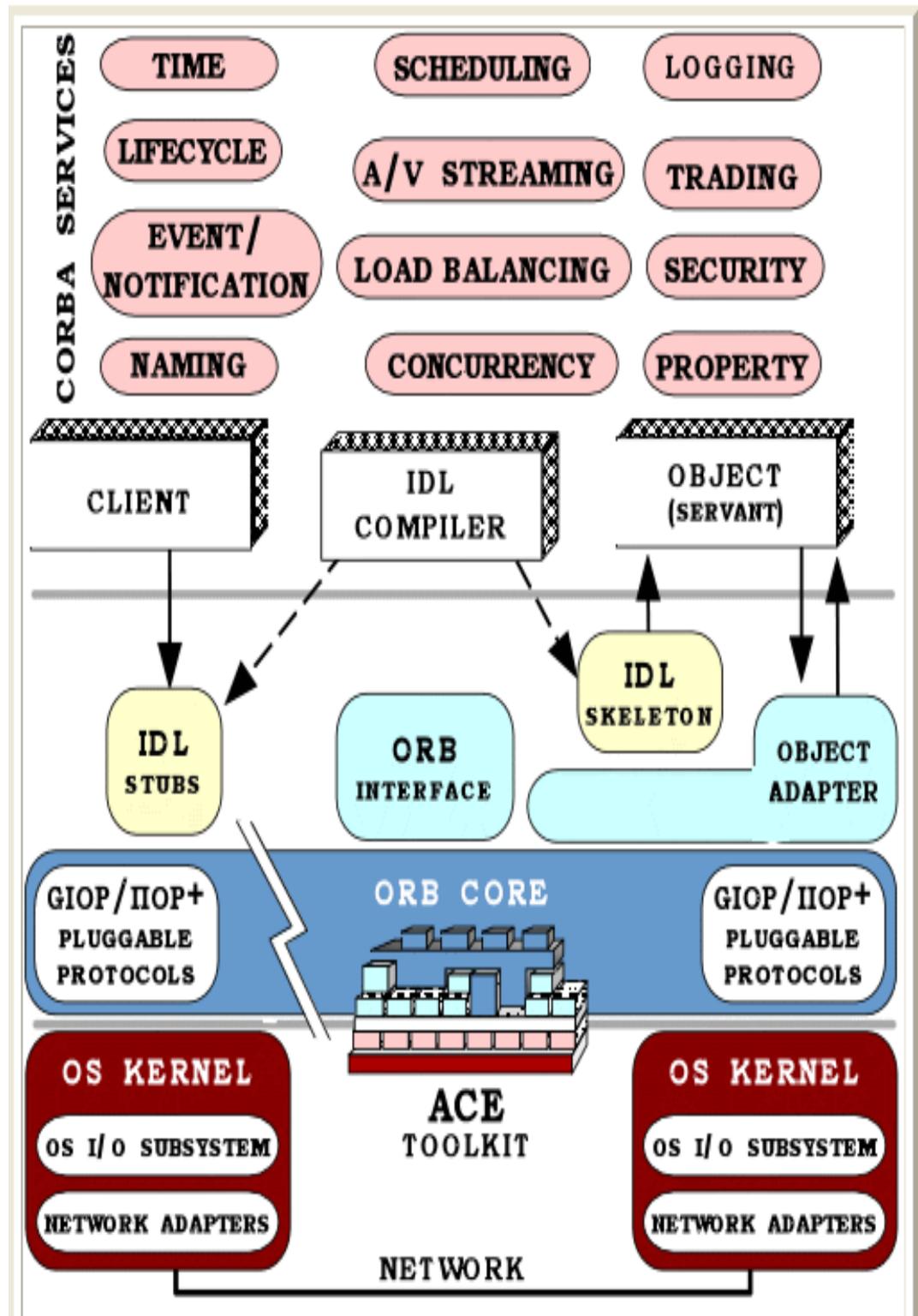
Tutkimusosuusanalyysissä selvisi, että Tuxedoa on käytetty hyvin vähän business-logiikan toteuttamiseen ja suurin osa palveluista on pelkkää sanomanvälitystä. Lisäksi selvisi, että kaikki kutsut Tuxedo-clienteista on toteutettu C-ohjelmointikielellä.

Korvaus tulisi olemaan suoraviivainen sellaisella sanomanvälitysohjelmistolla, joka tukee C tai C++ clientteja ja servereitä. Ulkoisten järjestelmien integraatiopajainta toteutetaan siten, että ulkoisille järjestelmille tarjotaan edelleen samat Tuxedo-palvelut, mutta väliin rakennetaan integraatiokerros, joka muuntaa Tuxedo-sanomat lankapuhelinverkon laskutusjärjestelmän vaatimaan muotoon ja myös muuntaa vastaussanomien takaisin Tuxedo-sanomiksi.

5.1 Korvaavan sanomanvälitysohjelmiston valinta

Tavoitteena oli löytää monipuolinen Open Source -sanomanvälitysohjelmisto, jonka suorituskyky on hyvä ja jolle saa myös tarvittaessa tukisopimuksen. Korvaavaksi sanomanvälitysohjelmistoksi valittiin TAO CORBA, joka on alun perin tehty Washingtonin yliopistossa. TAO CORBA on käytössä useissa eri maissa vaativissa sovelluksissa esimerkiksi Boeingin Earborne Early Warning & Control-järjestelmässä AEW&C, ja sille saa myös tarvittaessa tukisopimuksen (Members of the ACE and TAO Development Team 2010; ACE, TAO, and CIAO Success Stories 2011; OCI's Support Model for ACE and TAO 2011).

Sanomanvälitysohjelmiston ominaisuuksia on kuvattu kuviossa 3. Ohjelmisto tukee esimerkiksi kuormantasaus LOAD BALANCING, lokinkäsittely LOGGING ja nimipalvelu NAMING -palveluita. Tiedonsiirto clientin ja serverin välillä tapahtuu client stub-, ORB CORE-, OS KERNEL-, ORB INTERFACE- ja server skeleton -kerrosten kautta.



KUVIO 3. TAO CORBA -sanomanvälitysohjelmiston ominaisuuksia

5.1.1 Korvaavan sanomanvälitysohjelmiston toimintaperiaate

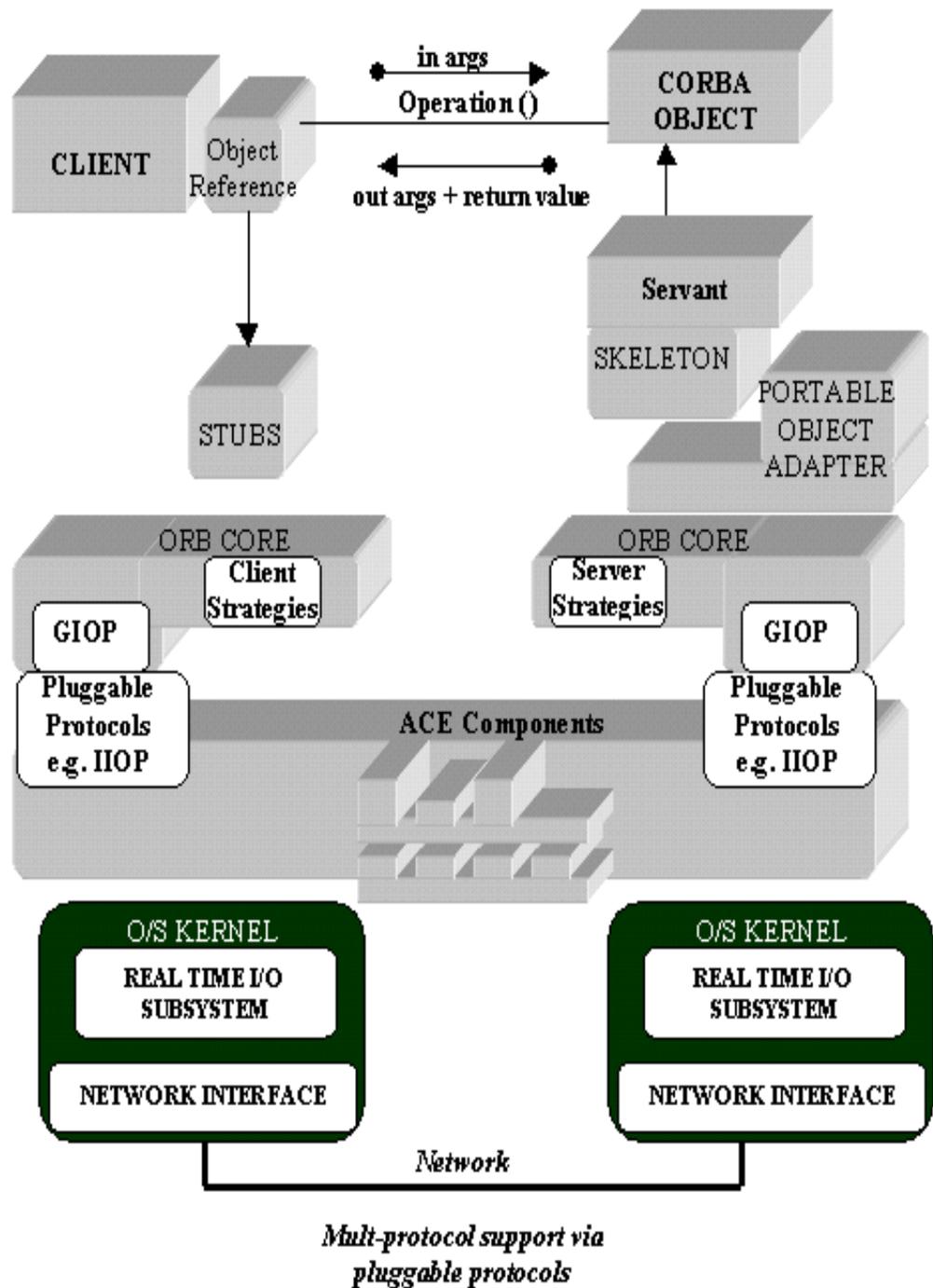
Clientin ja serverin välinen kutsurajapinta kuvataan rajapinnanmäärittyskielellä IDL Interface Definition Language (OMG IDL: Details 2012). Rajapinnan määrittäminen kirjoitetaan tiedostoon ja tiedosto annetaan parametriksi IDL-kääntäjälle, joka generoi määrittysten perusteella clientpuolen sovittimen stub ja serveripuolen sovittimen skeleton (CORBA® BASICS 2012).

Tämän jälkeen toteutetaan clientpuolen kutsut ja serveripuolen palvelut. Kun client- ja serveriosuudet on saatu valmiiksi, niin clientista voidaan tehdä C/C++-kielisiä funktiokutsuja serverillä oleviin palveluihin.

Serveri-ohjelma voi sijaita myös eri palvelimella kuin itse client-ohjelma. Kun serveri käynnistyy, niin se rekisteröi adapterinsa nimipalveluun, josta clientit voivat käydä hakemassa sen ja tämän avulla clientit voivat muodostaa yhteyden palvelimeen (An Overview of the CORBA Portable Object Adapter 2012).

Sanomanvälitysohjelmiston toimintaperiaate on kuvattu kuviossa 4. Serverillä on adapteri (portable object adapter), jonka se rekisteröi nimipalveluun. Client hakee serverin adapterin nimipalvelusta, ja muodostaa siihen viittauksen (Object Reference), jonka avulla client voi käyttää serverin palveluita (corba object). Tiedon siirto clientin ja serverin välillä tapahtuu client stub-, ORB CORE-, ACE Components-, OS KERNEL- ja server skeleton -kerrosten kautta. Client voi välittää serverin palvelukutsussa input parametreja (in args) ja output parametreja (out args), sekä lisäksi kutsuttava palvelu voi palauttaa clientille paluuarvon (return value). Sanomanvälitysohjelmisto tukee esimerkiksi IIOP- ja GIOP -protokollia.

TAO and other Components in a real-time Endsystem



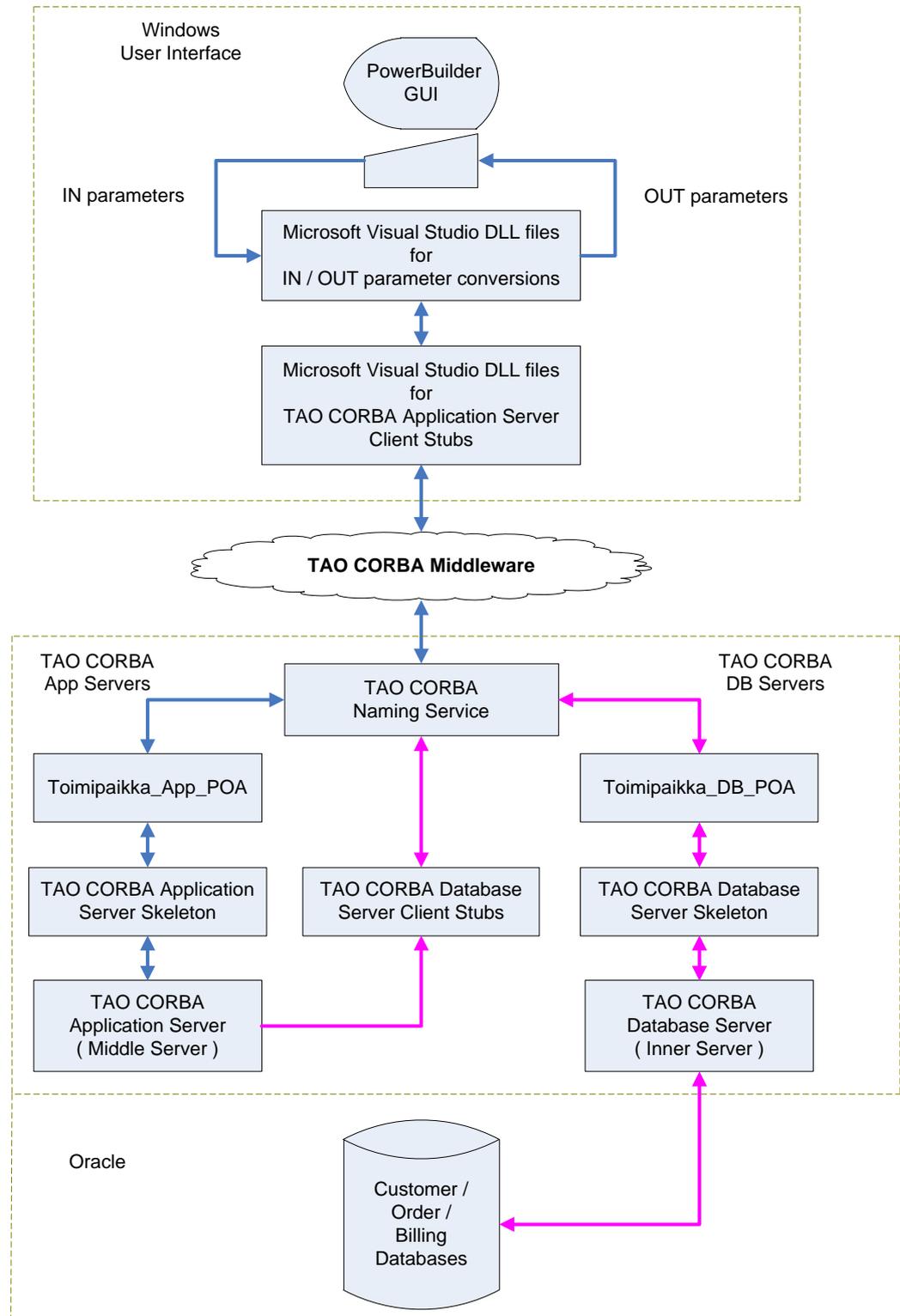
KUVIO 4. TAO CORBA -sanomanvälitysohjelmiston toimintaperiaate

5.1.2 Korvaavan sanomanvälitysohjelmiston testaus

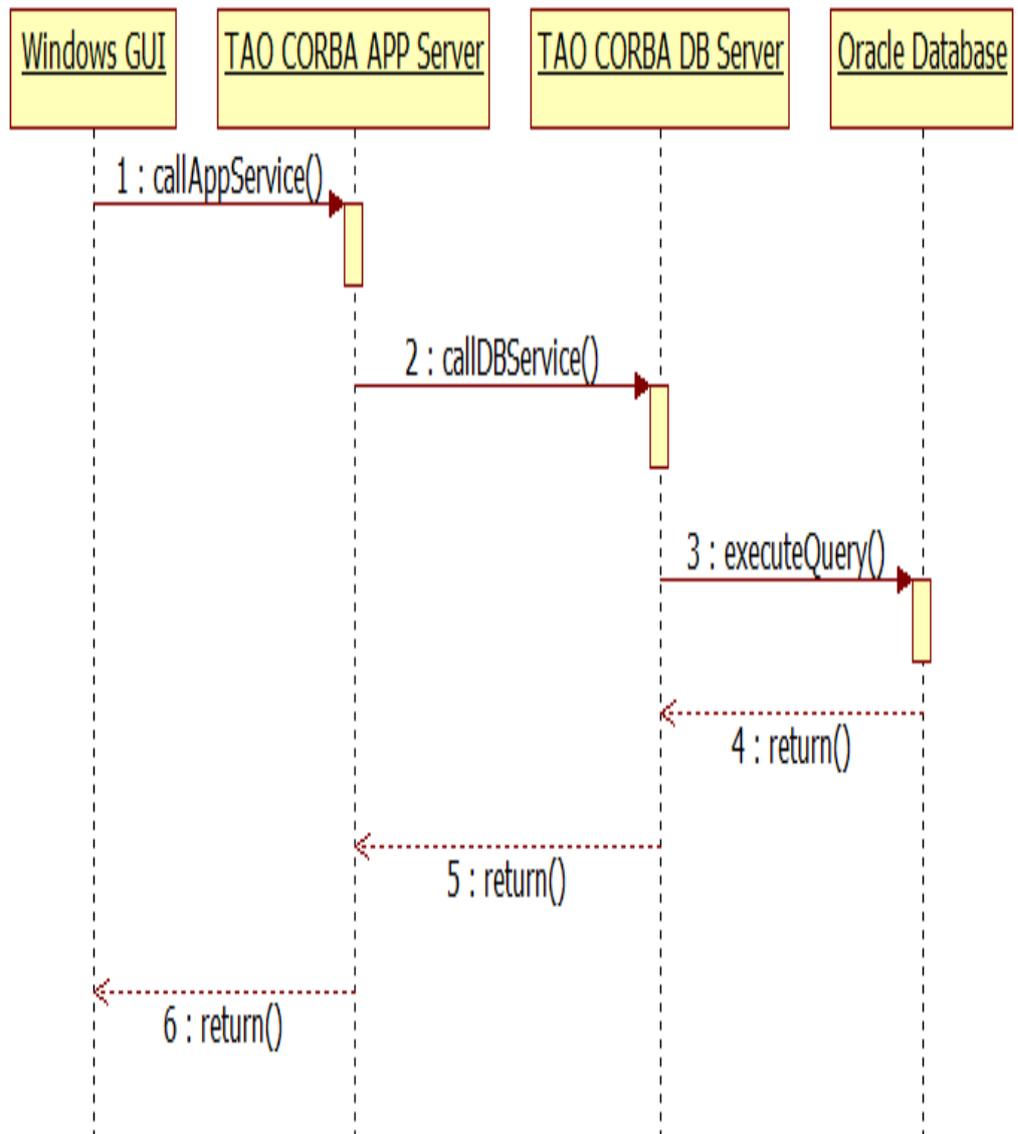
Testissä toteutettiin client, jolla pystyi kysymään serveriltä tietyn yrityksen osakekurssin arvoa. Tässä testissä client ja server käynnistettiin eri koneille ja lisäksi palvelinkoneelle käynnistettiin nimipalvelu. Nimipalvelua käytettiin myös kaikissa muissa testeissä, jotka on kuvattu seuraavissa luvuissa.

5.1.3 Monitasoisen arkkitehtuurin testaus korvaavalla sanomanvälitysohjelmistolla

Testin toimintaperiaate on kuvattu kuvioissa 5 ja 6. Testissä toteutettiin Power-Builder Windows TAO CORBA -client, jolla pystyi kysymään serveriltä tietyn toimipaikan tiedot postinumeron perusteella. Testissä käytetty arkkitehtuuri oli kolmitasoinen. Client lähetti kyselyn TAO CORBA -sovelluspalvelimelle, joka lähetti kyselyn edelleen TAO CORBA -kantapalvelimelle. Tietokantapalvelin teki kyselyn tietokantaan ja lähetti vastauksen sovelluspalvelimelle, joka lähetti vastauksen edelleen clientille.



KUVIO 5. Monitasoisen arkkitehtuuri testin toimintaperiaate



KUVIO 6. Monitaisen arkkitehtuuri testin sekvenssikaavio

5.1.4 Kuormantasauksen testaus korvaavalla sanomanvälitysohjelmistolla

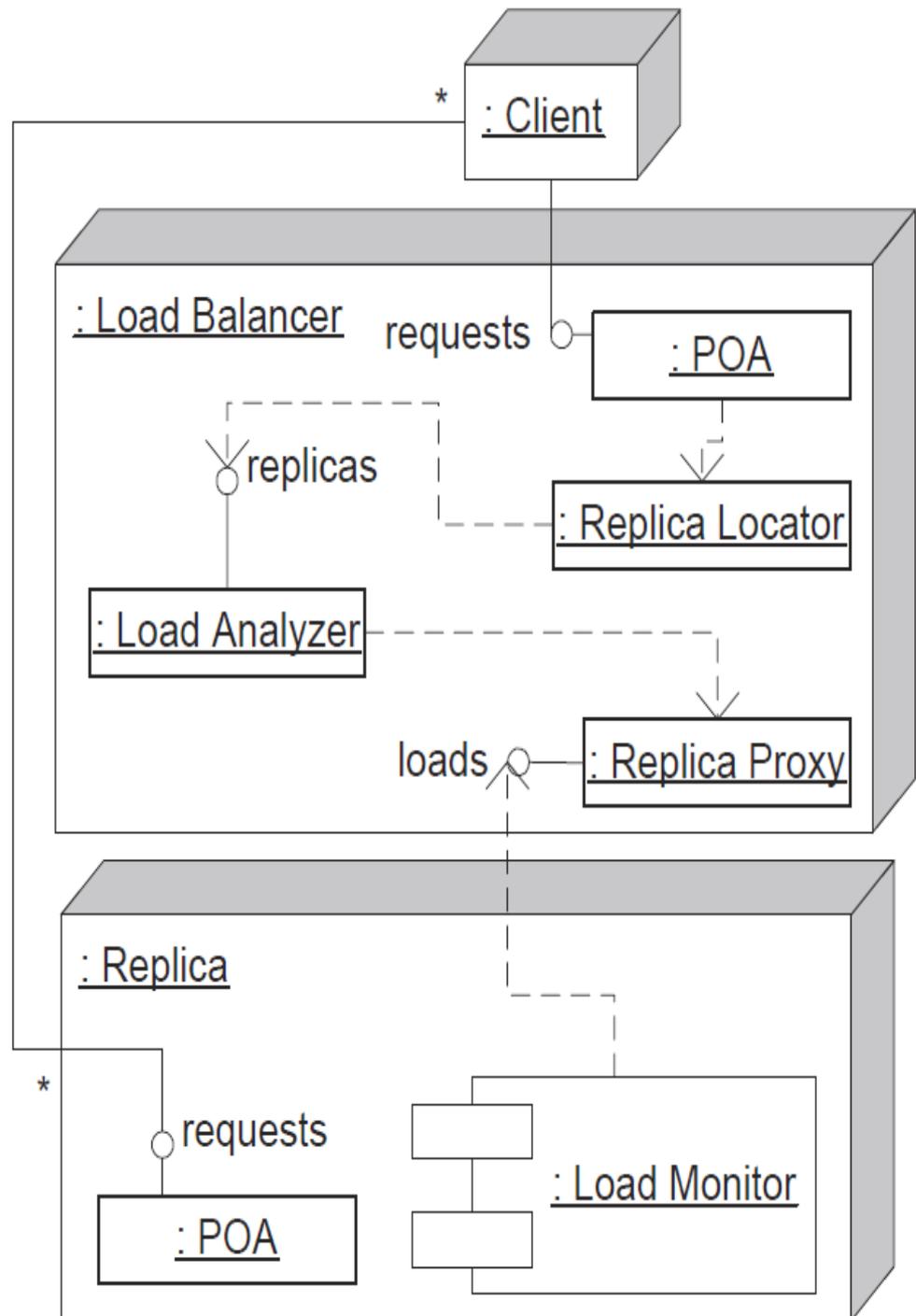
Luvussa 5.1.3 testatun monitasoisen arkkitehtuurin testausta laajennettiin kuormantasausominaisuuksilla. Kuormantasaus on tärkeä ominaisuus, jolla saadaan lisää skaalautuvuutta. Jos useilta clientelelta tulee samanaikaisia palvelupyynnöitä, niin kuormantasaus jakaa clienttien palvelupyynnöt kaikille niille palvelimille, joita ajetaan kuormantasauksessa. Kuormantasauksen ansiosta clienttien palvelupyynnöitä palvellaan rinnakkain. Kuormantasausta on käytetty tietyissä laskutusjärjestelmän Tuxedo-palveluissa, joten näihin palveluihin täytyy toteuttaa kuormantasaus myös korvaavalla sanomanvälitysohjelmistolla.

TAO CORBA tukee myös kuormantasausta, ja se on clientelelle läpinäkyvää, toisin sanoen kuormantasauksen käyttöönotto ei aiheuta mitään muutoksia clientohjelmistoon (The Design of an Adaptive CORBA Load Balancing Service 2001). TAO CORBA:n kuormantasaukseen tarvittavat ominaisuudet lisättiin sovelluspalvelimeen ja kantapalvelimeen.

Kuormantasauksessa tarvittava ohjelmakoodi on geneeristä, ja se ei sisällä mitään businesslogiikkaa, joten sen lisääminen palvelinten ohjelmakodeihin oli suoraviivaista ja nopeaa. TAO CORBA:n kuormantasaus käyttää myös LoadManager-nimistä apuohjelmaa, joka täytyi käynnistää palvelinkoneelle.

Testissä käynnistettiin neljä TAO CORBA -clienttia, kaksi sovelluspalvelinta ja kaksi kantapalvelinta. Clientit lähettivät 50 kyselyä palvelimille, ja testissä todettiin, että kyselyt ohjautuivat kaikille sovellus- ja kantapalvelimille ja sovelluspalvelimet lähettivät vastaukset virheettömästi kaikille clientelelle.

Kuormantasauksen toimintaperiaate on kuvattu kuviossa 7. Kuormantasauksessa ajettavat palvelimet muodostavat kuormantasausryhmän. Yksi ryhmän palvelin rekisteröi adapterinsa nimipalveluun, ja muut ryhmän palvelimet liittyvät tähän ryhmään. Kuormantasaaja (Load Balancer) tarkkailee palvelimien kuormituksia ja ohjaa clientin palvelupyynnön jollekin kuormantasauryhmän palvelimelle (Replica).



KUVIO 7. Kuormantasauksen toimintaperiaate

5.1.5 Asynkronisten sanomien testaus korvaavalla sanomanvälitysohjelmistolla

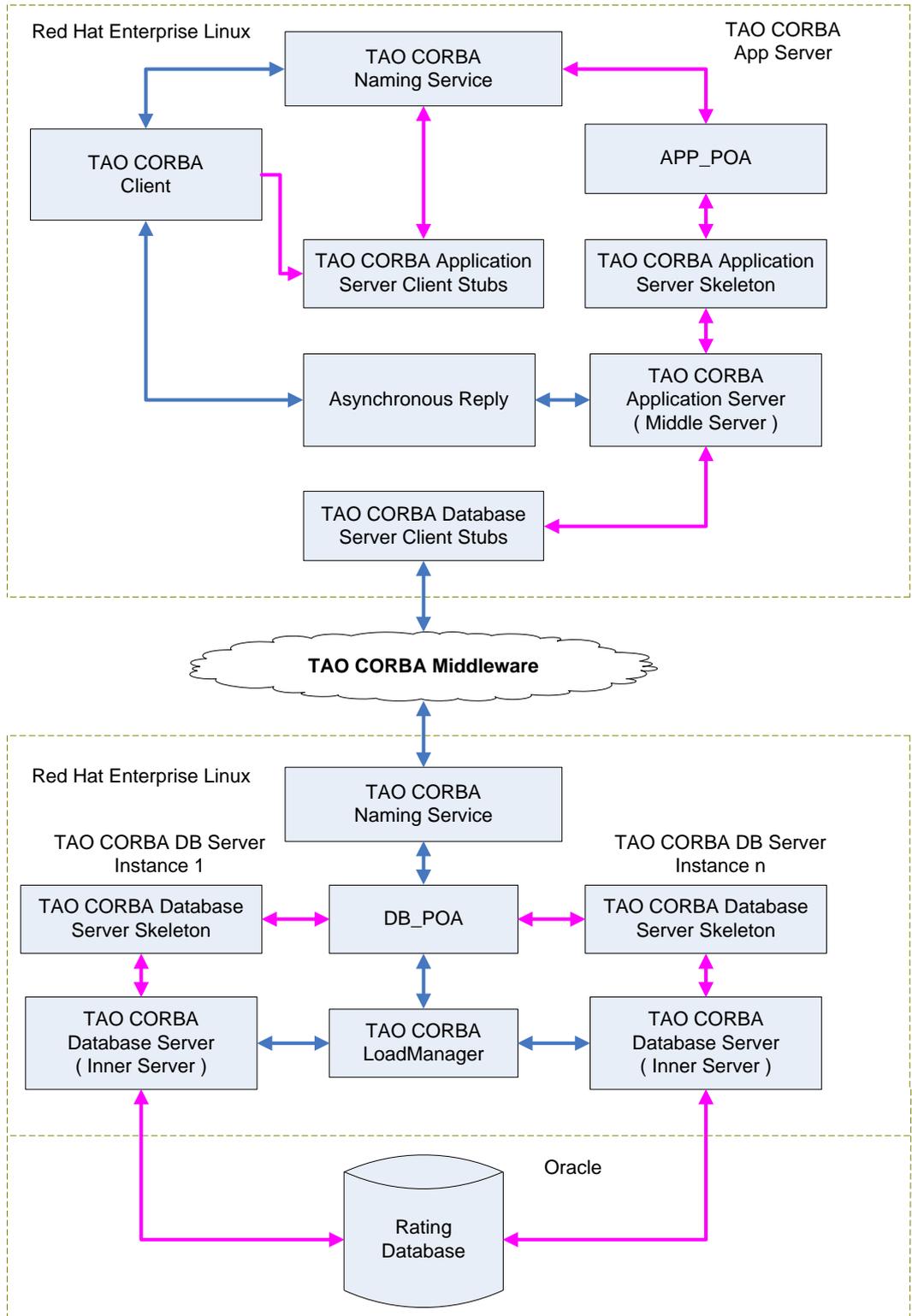
TAO CORBA tukee myös asynkronisiä palvelukutsuja, ja tämä on servereille läpinäkyvää, toisin sanoen asynkronisten palvelukutsujen käyttöönotto ei aiheuta muutoksia serveriohjelmistoon. Asynkronisiä palvelukutsuja on käytetty tietyissä laskutusjärjestelmän Tuxedo-clienteissa, joten näihin clientteihin täytyy toteuttaa vastaavat ominaisuudet myös korvaavalla sanomanvälitysohjelmistolla.

TAO CORBA:n asynkronisessa sanomanvälityksessä tarvittavat ominaisuudet lisättiin testissä käytettyyn clienttiin. Asynkronisessa sanomanvälityksessä tarvittava ohjelmakoodi on geneeristä, eikä se sisällä mitään businesslogiikkaa, joten sen lisääminen clientin ohjelmakoodeihin oli suoraviivaista ja nopeaa.

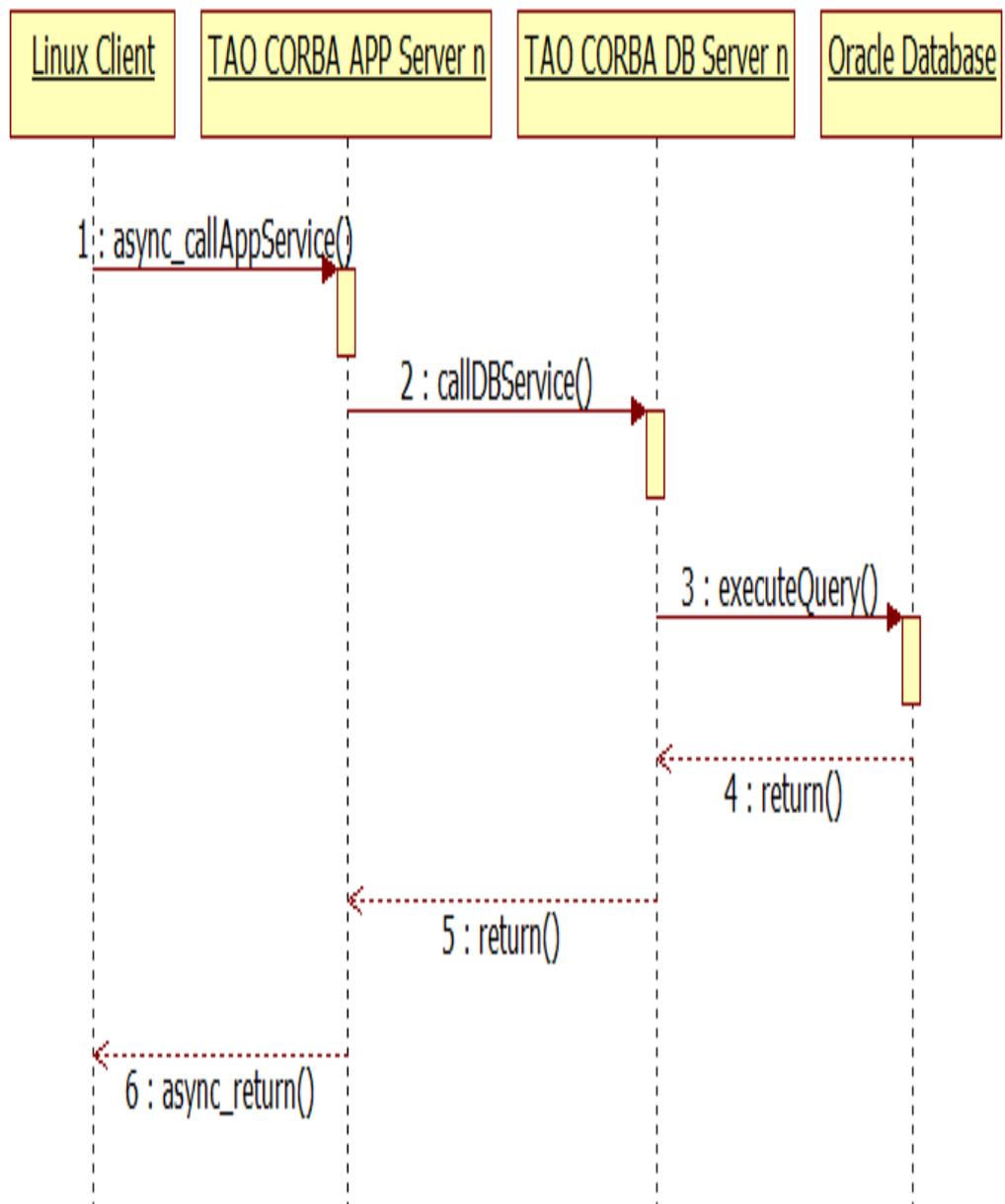
Testissä toteutettiin RedHat Enterprise TAO CORBA Linux client, sovelluspalvelin ja kantapalvelin. Testissä käytettiin kahta RedHat Enterprise Linux -palvelinta. Testin toimintaperiaate on kuvattu kuvioissa 8 ja 9.

Toiselle palvelimelle käynnistettiin TAO CORBA -client, neljä sovelluspalvelinta ja nimipalvelu ja toiselle palvelimelle neljä TAO CORBA -kantapalvelinta, nimipalvelu sekä TAO CORBA:n LoadManager. Client generoi joukon sanomia ja lähetti ne asynkronisina sanomina tasaisesti jokaiselle sovelluspalvelimelle. Sovelluspalvelimet lähettivät edelleen sanomat synkronisina kutsuina toisella palvelimella oleville kantapalvelimille, joita ajettiin TAO CORBA:n kuormantasauksen avulla.

Kantapalvelimet suorittivat tarvittavat kankakyselyt ja lähettivät vastaukset takaisin toisella palvelimella oleville sovelluspalvelimille. Sovelluspalvelimet lähettivät vastaukset takaisin edelleen clientille.



KUVIO 8. Asynkronisen arkkitehtuuri testin toimintaperiaate



KUVIO 9. Asynkronisen arkkitehtuuri testin sekvenssikaavio

5.1.6 IDL-tiedoston kääntäminen lähdekoodiksi

Kuviossa 10 on esimerkki TAO CORBA IDL-tiedostosta:

- Rajapinnan (interface) nimi on Middle.
- Interfacen sisällä on yksi metodi, jonka nimi on getMunicipality.
- Metodille välitetään kutsussa kaksi string tyyppistä parametria
 - parametrin tyyppi voi olla
 - in = input
 - out = output
 - inout = input ja output.
- Metodi palauttaa yhden string tyyppisen parametrin.

```

1 interface Middle
2 {
3     // Fetches municipality by postal code
4     string getMunicipality (in string postal_code, in string language_code);
5 };

```

CUVIO 10. Esimerkki TAO CORBA IDL-tiedostosta

Tallennetaan IDL-tiedosto esimerkiksi middle.idl nimellä, ja käännetään se käyttöjärjestelmässä IDL -kääntäjällä seuraavalla komennolla:

- tao_idl -GI middle.idl

Käännöksen tuloksena muodostuu clientin stubi, serverin skeleton ja serverin interface tiedostot, joihin toteutetaan getMunicipality metodi.

Clientin stubi

- middleC.cpp
- middleC.h
- middleC.inl.

Serverin skeleton

- middleS.cpp
- middleS.h
- middleS.inl.

Serverin interface tiedostot

- middleI.cpp
- middleI.h.

Kääntäjän muodostamassa inl-tiedostossa on C++-ohjelmointikielen inline-koodia. Kuviossa 11 on esimerkki IDL-kääntäjän generoimasta middleI.cpp tiedostosta ja kuviossa 12 esimerkki IDL-kääntäjän generoimasta middleI.h tiedostosta.

```

21//      Nashville, TN
22//      USA
23//      http://www.isis.vanderbilt.edu/
24//
25// Information about TAO is available at:
26//      http://www.cs.wustl.edu/~schmidt/TAO.html
27
28// TAO_IDL - Generated from
29// be/be_codegen.cpp:1179
30
31#include "middleI.h"
32
33// Implementation skeleton constructor
34Middle_i::Middle_i (void)
35{
36}
37
38// Implementation skeleton destructor
39Middle_i::~Middle_i (void)
40{
41}
42
43char * Middle_i::getMunicipality (
44    const char * postal_code,
45    const char * language_code
46 )
47{
48    // Add your implementation here
49}

```

KUVIO 11. Esimerkki IDL-kääntäjän generoimasta middleI.cpp tiedostosta

```

29// be/be_codegen.cpp:1116
30
31#ifndef MIDDLEI_H_
32#define MIDDLEI_H_
33
34#include "middleS.h"
35
36#if !defined (ACE_LACKS_PRAGMA_ONCE)
37#pragma once
38#endif /* ACE_LACKS_PRAGMA_ONCE */
39
40class Middle_i
41 : public virtual POA_Middle
42 {
43 public:
44     // Constructor
45     Middle_i (void);
46
47     // Destructor
48     virtual ~Middle_i (void);
49
50     virtual
51     char * getMunicipality (
52         const char * postal_code,
53         const char * language_code
54     );
55 };
56#endif /* MIDDLEI_H_ */

```

KUVIO 12. Esimerkki IDL-kääntäjän generoimasta middleI.h tiedostosta

5.1.7 Make Project Creator työkaluohjelman käyttäminen

Tällä työkaluohjelmalla voi generoida makefile-tiedostoja eri kääntäjäympäristöihin. Ohjelmalle kirjoitetaan parametriksi tiedosto, jossa kuvataan kaikki projektiin kuuluvat lähdetiedostot. Työkaluohjelma etsii ajohakemistosta tiedostoja, joiden loppuliite on mpc, ja se muodostaa makefile-tiedostojen sisällön näiden tiedostojen perusteella.

Kuviossa 13 on esimerkki ohjelman parametritiedostosta. Serverin koodi on kirjoitettu middle_server.cpp tiedostoon, ja parametritiedosto on tallennettu middle.mpc tiedostoon.

```

1 project(*Middle_Server): messaging, taoexe, portableserver {
2   idlflags += -Wb
3   after += *Inner_Server
4
5   IDL_Files {
6     middle.idl
7   }
8
9   Source_Files {
10    middleS.cpp
11    middleC.cpp
12    innerS.cpp
13    innerC.cpp
14    middleI.cpp
15    middle_server.cpp
16  }
17 }

```

KUVIO 13. Esimerkki MPC-työkaluohjelman parametritiedostosta

Makefile-tiedoston generointi käynnistetään käyttöjärjestelmässä seuraavalla komennolla:

- perl \$ACE_ROOT/bin/mwc.pl -type make
- \$ACE_ROOT on hakemisto, johon ACE_WRAPPERS on asennettu

5.1.8 TAO CORBA -serverin toimintaperiaate

Serverin tarjoamien metodien toteutukset kirjoitetaan interface tiedostoon. Esimerkiksi lukuun 5.1.6 viitaten `getMunicipality` metodin toteutus kirjoitetaan `middleI.cpp` tiedostoon ja metodin esittely `middleI.h` tiedostoon. Serveri rekisteröi itsensä nimipalveluun yksilöidyllä adapterinimellä. Adapterin rekisteröinti tehdään Portable Object Adapterina POA (ORB Basics 2012).

Esimerkiksi sivulla 14 kuviossa 5 sovelluspalvelin on rekisteröinyt itsensä nimipalveluun nimellä `Toimipaikka_App_POA`, ja tietokantapalvelin nimellä `Toimipaikka_DB_POA`. Vanhemmissa CORBA-versioissa adapterin rekisteröinti tehtiin Basic Object Adapterina BOA, mutta tämä tekniikka ei ole siirrettävä eri CORBA-versioiden välillä (ORB Basics 2012).

Nimipalveluun rekisteröinnin jälkeen serveri kutsuu run metodia, joka odottaa clienttien yhteydenottoja. Kun clientti ottaa yhteyden serveriin, niin sen kutsu palvelee omassa säikeessä, ja serverin pääsäie jatkaa clienttien yhteydenottojen kuuntelua. Tämä on TAO CORBA -serverin peruskonfiguraatio.

Konfiguraation muutoksella TAO CORBA -serveri saadaan toimimaan myös monisäikeisesti, jolloin serveri luo uuden säikeen jokaiselle clientin kutsulle ja kutsu palvelee omassa säikeessä. Serverin koodi täytyy kirjoittaa tässä konfiguraatiossa säieturvallisesti, eli globaalien ja staattisten muuttujien käsittely täytyy synkronoida.

Serveriin linkitetään linkkausvaiheessa mukaan IDL-kääntäjän generoimat skeleton-tiedostot. Kuviossa 14 on esimerkki TAO CORBA -serverin lähdekoodista.

```

30 // Register the servant with the RootPOA, obtain its object
31 // reference, stringify it, and write it to stdout.
32 PortableServer::ObjectId_var oid = poa->activate_object(servant.in());
33 obj = poa->id_to_reference( oid.in() );
34
35 CORBA::String_var ior = orb->object_to_string( obj.in() );
36 std::cout << ior.in() << std::endl;
37
38 CosNaming::Name name_mi (1);
39 name_mi.length (1);
40
41 name_mi[0].id = CORBA::string_dup ("Toimipaikka_App_POA");
42 naming_context->bind (name_mi, obj.in() );
43
44 // Accept requests from clients.
45 orb->run();
46
47 poa->destroy (1, 1);
48 orb->destroy();
49
50 return 0;
51 }
52 catch(const CORBA::Exception& ex) {
53     std::cerr << "CORBA exception: " << ex << std::endl;
54 }
55
56 return 1;
57 }

```

KUVIO 14. Esimerkki TAO CORBA -serverin lähdekoodista

5.1.9 TAO CORBA -clientin toimintaperiaate

TAO CORBA -client hakee käynnistyessään niiden serverien adapterit nimipalvelusta, joiden palveluita sen koodissa kutsutaan. Sivulla 14 kuviossa 5 Windows PowerBuilder -client on hakenut nimipalvelusta sovelluspalvelimen Toimipaikka_App_POA adapterin, jotta se voi kutsua sovelluspalvelimen palvelua.

Kuviossa 5 Red Hat Enterprise Linux -sovelluspalvelin on hakenut nimipalvelusta tietokantapalvelimen Toimipaikka_DB_POA adapterin, jotta se voi kutsua tietokantapalvelimen palvelua. TAO CORBA -clienttiin linkitetään linkkausvaiheessa mukaan IDL-kääntäjän generoimat stub-tiedostot niiden serverien osalta, joiden palveluita sen koodissa kutsutaan.

Kuvion 5 Windows PowerBuilder -clienttiin linkitetään mukaan sovelluspalvelimen client stub. Kuvion 5 Red Hat Enterprise Linux -sovelluspalvelimeen linkitetään mukaan tietokantapalvelimen client stub.

5.1.10 TAO CORBA -nimipalvelu

TAO CORBA -nimipalvelu käynnistetään käyttöjärjestelmässä seuraavalla komennolla:

- `$TAO_ROOT/orbsvcs/Naming_Service/Naming_Service -ORBEndPoint iiop://10.0.2.15:12345`
- `$TAO_ROOT` on hakemisto, johon TAO on asennettu
- nimipalvelulle annetaan käynnistyksessä parametrina portin numero, jota se kuuntelee ja koneen IP-osoite, jolle se käynnistetään
 - Esimerkissä on käytetty porttia 12345 ja IP-osoitetta 10.0.2.15.

Nimipalvelulle voidaan antaa käynnistysvaiheessa myös muita parametreja, esimerkiksi:

- `-d`, jolla säädetään debug tulostusten tasoa.

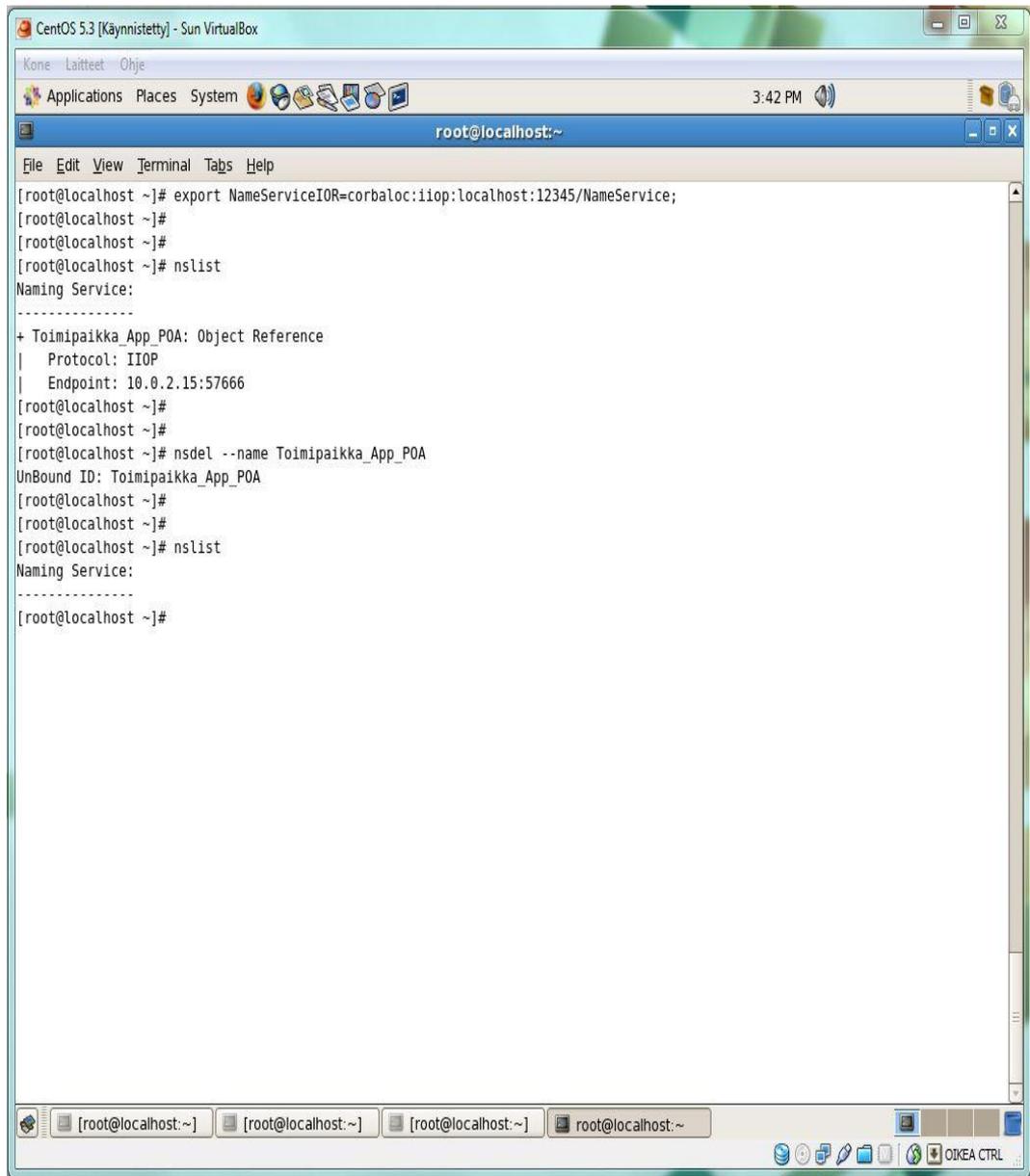
Nimipalveluun rekisteröidyt adapterit voidaan tulostaa käyttöjärjestelmässä seuraavalla komennolla:

- `$TAO_ROOT/utills/nslist/nslist`
- ennen komennon käynnistämistä täytyy asettaa `NameServiceIOR-` ympäristömuuttuja, jolla kerrotaan nimipalvelun IP-osoite ja porttinumero.

Nimipalveluun rekisteröity adapteri voidaan poistaa käyttöjärjestelmässä seuraavalla komennolla:

- `$TAO_ROOT/utills/nslist/nsdel --name Toimipaikka_App_POA`
- ennen komennon käynnistämistä täytyy asettaa `NameServiceIOR-` ympäristömuuttuja, jolla kerrotaan nimipalvelun IP-osoite ja porttinumero
- komennolle annetaan parametrina poistettavan adapterin nimi
 - Esimerkissä on käytetty adapterin nimeä `Toimipaikka_App_POA`.

Kuviossa 15 on esimerkki nimipalvelun adapterien tulostamisesta ja yhden adapterin poistamisesta.



```
CentOS 5.3 [Käynnistetty] - Sun VirtualBox
Kone Laitteet Ohje
Applications Places System 3:42 PM
root@localhost:~
File Edit View Terminal Tabs Help
[root@localhost ~]# export NameServiceIOR=corbaloc:iiop:localhost:12345/NameService;
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# nslslist
Naming Service:
-----
+ Toimipaikka_App_POA: Object Reference
| Protocol: IIOP
| Endpoint: 10.0.2.15:57666
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# nsdel --name Toimipaikka_App_POA
UnBound ID: Toimipaikka_App_POA
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# nslslist
Naming Service:
-----
[root@localhost ~]#
```

KUVIO 15. Esimerkki nimipalvelun adapterien tulostamisesta ja yhden adapterin poistamisesta

5.2 Integraatio-ohjelmiston valinta

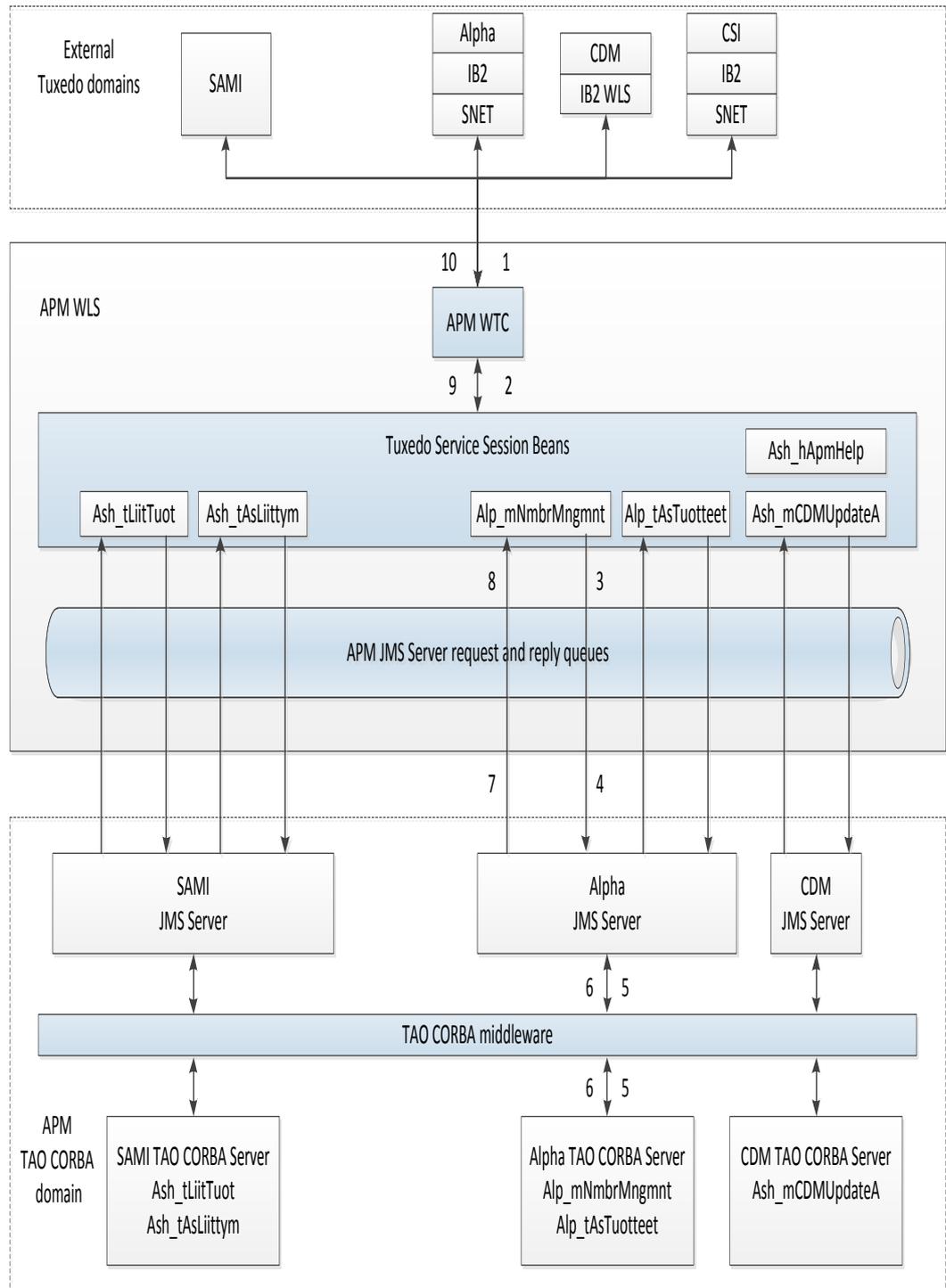
Tavoitteena oli löytää ohjelmisto, joka tukee Tuxedo-integraatioiden lisäksi myös muita integraatioita ja jonka suorituskyky on hyvä. Integraatio-ohjelmistoksi valittiin Oracle WebLogic Server, jonka nimi oli aikaisemmin BEA WebLogic Server (Oracle WebLogic Server Overview 2012). Serveri tukee monipuolisia integraatiotekniikoita, esimerkiksi Tuxedo ja JMS-sanomajonoja.

5.2.1 Integraatio-ohjelmiston toimintaperiaate

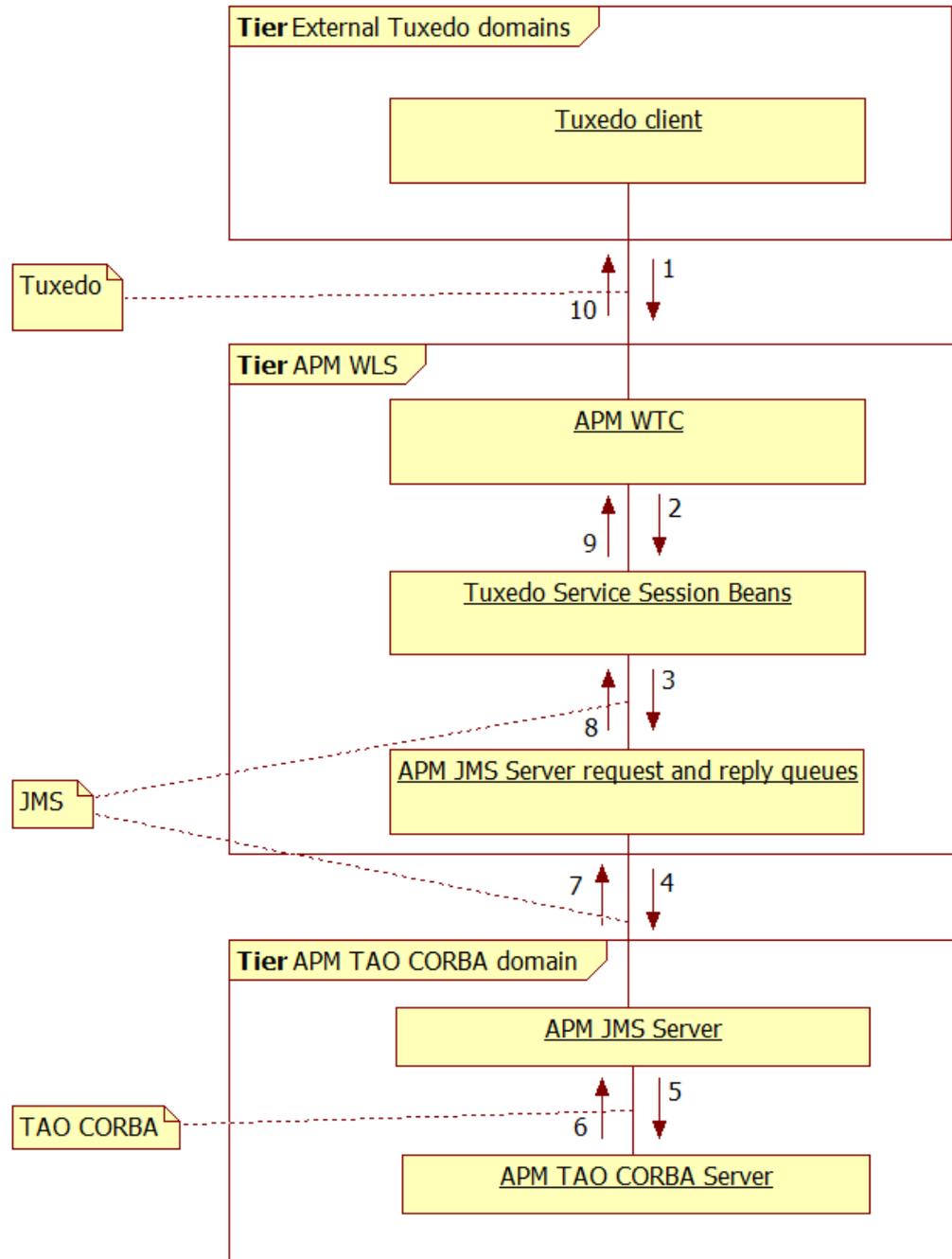
WebLogic Serverin Tuxedo Connector vastaanottaa ulkoisen järjestelmän Tuxedo-sanoman ja käynnistää Java-luokan, jossa palvelukutsu suoritetaan (1 ja 2). Java-luokka muuntaa Tuxedo-sanoman XML-sanomaksi ja kirjoittaa sen JMS-sanomajonoon (3).

Laskutusjärjestelmän JMS-serveri lukee pyyntösanoman JMS-sanomajonosta, muuntaa sen, ja kutsuu TAO CORBA -serverin palvelua, joka suorittaa varsinaisen palvelun (4 ja 5). TAO CORBA -serveri palauttaa vastauksen laskutusjärjestelmän JMS-serverille, joka muuntaa sen, ja kirjoittaa sen XML-sanomana JMS-sanomajonoon (6 ja 7).

WebLogic Serverin Java-luokka lukee vastauksen JMS-sanomajonosta, muuntaa sen Tuxedo-sanomaksi, ja palauttaa vastauksen ulkoisen järjestelmän clientille (8, 9 ja 10). Integraatio-ohjelmiston toimintaperiaate on kuvattu kuvioissa 16 ja 17.



KUVIO 16. Integraatio-ohjelmiston arkkitehtuuri



KUVIO 17. Integraatio-ohjelmiston collaboration model

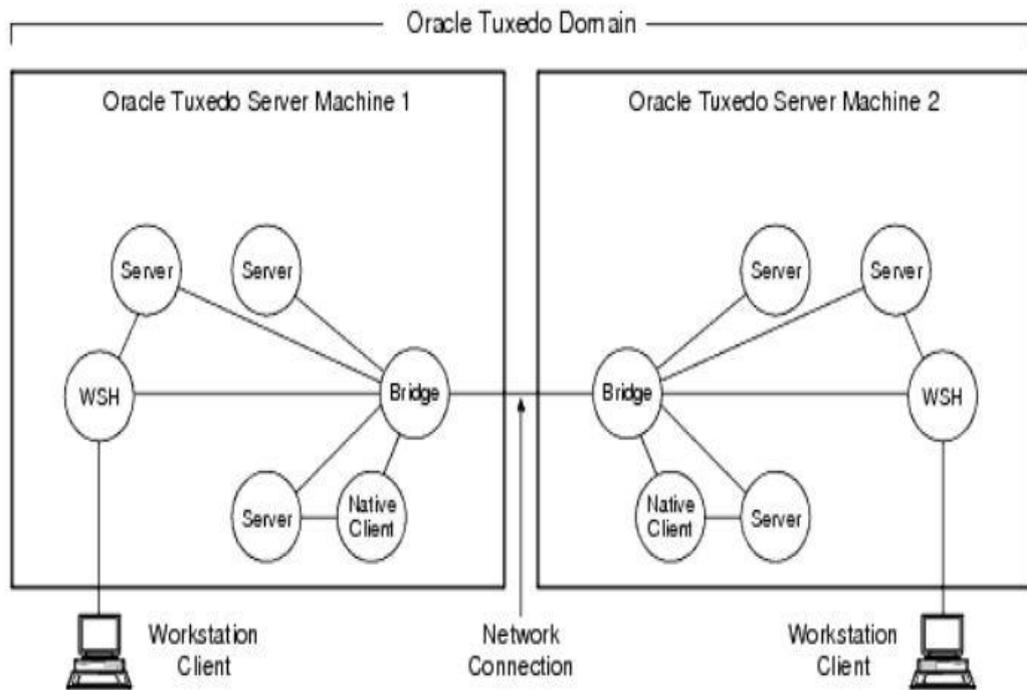
5.2.2 Tuxedo-domainin kuvaus

Tuxedo-domain on joukko Tuxedo-järjestelmä-, client- ja server-prosesseja, joita hallinnoidaan yhtenä kokonaisuutena yhden Tuxedo-konfiguraatitiedoston avulla. Tuxedo-domain koostuu monista järjestelmäprosesseista, yhdestä tai monesta client-prosessista, yhdestä tai monesta serveri-prosessista ja yhdestä tai monesta tietoliikenneverkkoon kytketystä palvelimesta. (Introducing Oracle Tuxedo ATMI 2008.)

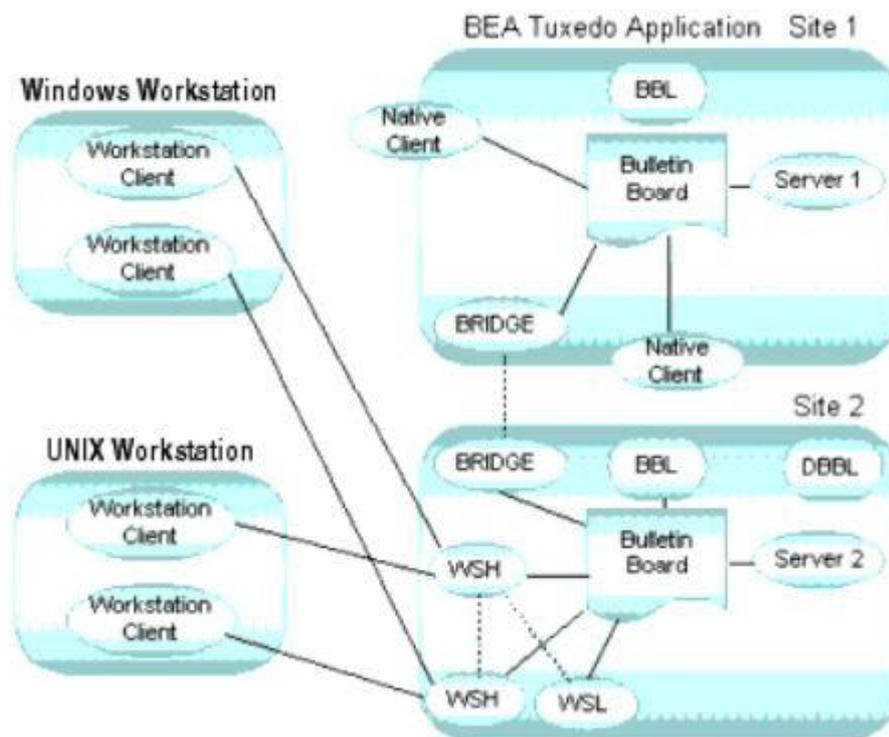
Tuxedo-domainilla on palvelimen sisäinen jaettu muistialue, jota kutsutaan Bulletin Boardiksi. Bulletin Boardiin on rekisteröity kaikki ne palvelut, joita Tuxedo-domainin serverit tarjoavat clienteleille. Koska Bulletin Board on palvelimen muistissa, niin vain samalla palvelimella olevilla clienteleilla on siihen suora pääsy, ja näitä clienteleja kutsutaan Tuxedo Native Clienteiksi.

Tuxedo-domainin palveluihin voi muodostaa yhteyden domainin sisältä Tuxedo Native Clientista, toisen Tuxedo-Domainin Native Clientista tai domainin ulkopuolella olevasta Tuxedo Workstation Clientista. Tuxedo Workstation Clientilla ei ole omaa Tuxedo-domainia, vaan kyseessä on kevyt työasemaversio, joka vaatii toimiakseen täyden Tuxedo-domain asennuksen palvelimen puolelle.

Tuxedo Workstation Client muodostaa yhteyden Tuxedo-domainin Workstation Listener -prosessiin WSL, joka ohjaa yhteyden kutsujalle läpinäkyvästi Workstation Handler -prosessille WSH. WSH ohjaa yhteydenoton Bulletin Boardin avulla oikealle serverille, jonka palvelua Tuxedo Workstation Client on kutsunut. Kuviossa 18 on kuvattu Tuxedo-domainin korkean tason kuvaus ja kuviossa 19 miten WSL ja WSH ohjaavat Tuxedo Workstation Clientin yhteydenoton oikealle serverille.



KUVIO 18. Tuxedo-domainin korkean tason kuvaus



KUVIO 19. Tuxedo Workstation Clientin yhteydenotto Tuxedo-serveriin

5.2.3 Integraatio-ohjelmiston komponentit

Tässä luvussa on kuvattu integraatio-ohjelmiston eri komponentit ja niiden tehtävät. Integraatio-ohjelmisto tarjoaa ulkoisille Tuxedo-domaineille Tuxedo kutsurajapinnan, jonka välityksellä ulkoiset Tuxedo-domainit voivat kutsua laskutusjärjestelmän palveluita.

- **External Tuxedo domains:** Ulkoiset Tuxedo-domainit, jotka kutsuvat laskutusjärjestelmän palveluita.
- **WLS:** Laskutusjärjestelmän WebLogic Server instanssi.
- **TAO CORBA domain:** Laskutusjärjestelmän TAO CORBA -domain, joka suorittaa varsinaiset palvelut.
- **WTC:** WebLogic Tuxedo Connector, joka muodostaa gatewayn ulkoisten Tuxedo-domainien kanssa ja tarjoaa niille Tuxedo-kutsurajapinnan.
- **Tuxedo Service Session Beans:** Java-luokat, jotka suorittavat Tuxedo FML32 tai Tuxedo CARRAY <-> JMS XML muunnoksen.
- **WLS JMS Server request and reply queues:** JMS-server, jossa jonot sijaitsevat.
- **JMS Server:** Laskutusjärjestelmän JMS-serveri, joka vastaanottaa pyynnön XML:nä JMS request -jonoon, lähettää pyynnön CORBA-sanomana TAO CORBA -serverille, vastaanottaa CORBA-vastauksen ja palauttaa vastauksen XML:nä reply jonoon.
- **TAO CORBA Server:** Laskutusjärjestelmän TAO CORBA -serveri, joka vastaanottaa pyynnön CORBA-sanomana, suorittaa palvelun ja palauttaa vastauksen CORBA-sanomana.

5.2.4 Integraatio-ohjelmiston testaus

Ohjelmisto testattiin seuraavilla testeillä:

- yksikkötestaus
- Tuxedo-rajapinnan testaus
- JMS-rajapinnan testaus
- integraatiotestaus.

Yksikkötestauksessa yksittäinen palvelu testattiin siten, että siitä tehtiin itsenäinen sovellus. Sovelluksen pääohjelmassa muodostettiin Tuxedo-sanoma, ja sanoma välitettiin parametrina testattavalle palvelulle. Palvelu teki Tuxedo-sanomalle XML-muunnoksen, muodosti kovakoodatun XML-vastaussanomaa, parseroi sen, ja lopuksi muunsi XML-vastaussanomaa Tuxedo-sanomaksi ja palautti sen pääohjelmalle. Tällä menetelmällä palvelu saatiin testattua ilman integraatiota ulkoiseen järjestelmään ja laskutusjärjestelmään.

Tuxedo-rajapinta testattiin siten, että laskutusjärjestelmän Tuxedo-clientilla välitettiin palvelukutsu palvelulle. Palvelu palautti kovakoodatun vastaussanomaa clientille kuten yksikkötestissäkin.

JMS-rajapinta testattiin siten, että laskutusjärjestelmän JMS-serveri otettiin testiin mukaan, ja laskutusjärjestelmän Tuxedo-clientilla välitettiin kutsu palvelulle. Palvelu välitti kutsun JMS-serverille, joka välitti kutsun edelleen TAO CORBA -serverille. TAO CORBA -serveri suoritti varsinaisen palvelun, ja lopulta vastaussanoma palautui takaisin samaa reittiä laskutusjärjestelmän Tuxedo-clientille. Tällä menetelmällä varmistettiin koko integraation toimivuus päästä päähän laskutusjärjestelmän omalla testiohjelmalla.

Integraatiotestauksessa palvelu testattiin siten, että palvelukutsu välitettiin ulkoisen järjestelmän Tuxedo-clientista palvelulle. Tällä menetelmällä varmistettiin koko integraation toimivuus päästä päähän ulkoisen järjestelmän todellisella Tuxedo-clientilla.

5.2.5 Tuxedo-sanomien reititys

Alpha ja CSI ovat Tuxedo Workstation Client tyyppisiä clientteja, eli järjestelmillä ei ole omaa Tuxedo-domainia. Koska WebLogic Serverin Tuxedo Connectoriin pystyy ottamaan yhteyden vain Tuxedo-domainista tai toisesta WebLogic Server Tuxedo Connectorista, niin Alpha- ja CSI-järjestelmien kutsut piti kierrättää reitittävän Tuxedo-domainin kautta.

Reitittävä Tuxedo-domain muuntaa Tuxedo-sanomat Tuxedo Domains protokollamuotoon, ja välittää sanomat laskutusjärjestelmän WTC:lle. SNET-järjestelmä valittiin reitittäväksi Tuxedo-domainiksi, ja domainin konfiguraatioon tehtiin reititykseen tarvittavat muutokset. Reitityksen käyttöönotto ei vaatinut mitään muutoksia SNET-järjestelmän ohjelmistoon, eli kyseessä oli pelkkä konfiguraatiomuutos.

5.2.6 Palomuurin aukipitäminen ja ulkoisten Tuxedo-domainien tunnistus

Käytössä olevassa WebLogic Server -versiossa 8.1 ei ole niin sanottua keeplive-toimintoa, joka pitäisi socket-yhteydet aktiivisina, vaikka niiden käytössä olisi pidempiäkin taukoja. Tämän takia integraatio-ohjelmistoon toteutettiin WLS:n sisäinen palvelu, jota IB2-järjestelmän WebLogic Server Tuxedo-client voi kutsua ajastetusti, ja tällä menetelmällä palomuuuri saadaan pidettyä auki.

Integraatio-ohjelmistoon toteutettiin myös Tuxedo-client, joka kutsuu ajastetusti IB2-järjestelmän WebLogic Server -palvelua. Tällä menetelmällä palomuuuri saadaan pidettyä auki myös toiseen suuntaan.

WTC tunnistaa ulkoiset Tuxedo-domainit niiden domain-nimen perusteella. Domainien nimet luodaan uusina käyttäjätunnuksina WebLogic Serveriin sen konsolikäyttöliittymän kautta, ja käyttäjätunnusten salasana voidaan keksiä itse.

5.2.7 JMS XML request / reply -sanomien kohdistus ja poikkeustilanteiden käsittely

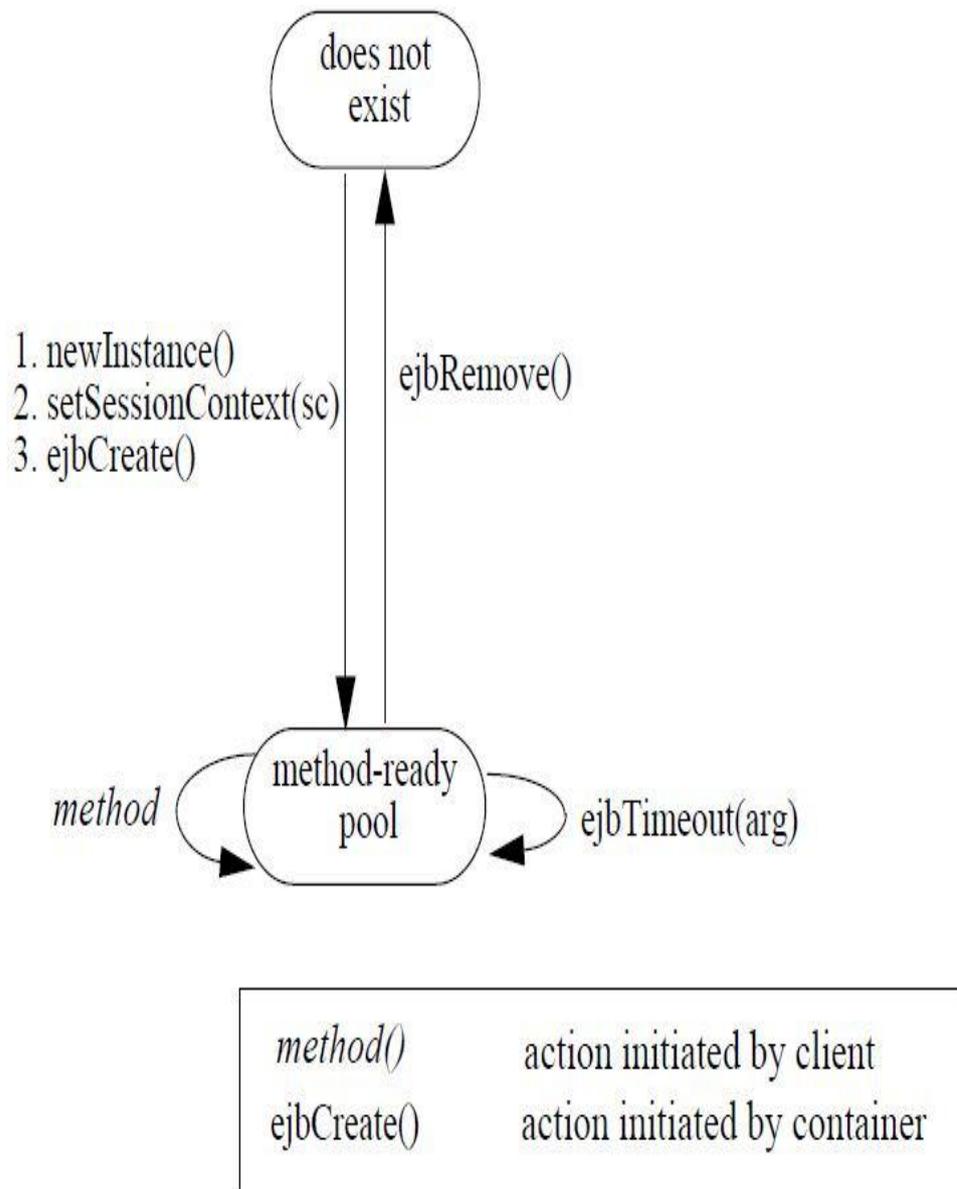
Tuxedo Service Session Beanit odottavat laskutusjärjestelmän JMS-serveriltä vastaussanomaa, jossa JMSCorrelationID = pyyntösanomien JMSMessageID. Poikkeuksena Ash_hApmHelp, joka on APM WLS:n sisäinen palvelu. Jos integraatio-sovelluksessa tapahtuu poikkeus, esimerkiksi kirjaimia sisältävä merkkijono yritetään muuntaa luvuksi, niin palvelun kutsujalle palautetaan laskutusjärjestelmän yleinen tekninen virhekoodi ja lyhyt tekstimuotoinen kuvaus virheestä. Virheen tarkempi syy tulostetaan integraatio-sovelluksen lokitiedostoon.

5.2.8 Stateless session bean Java-luokan toimintaperiaate

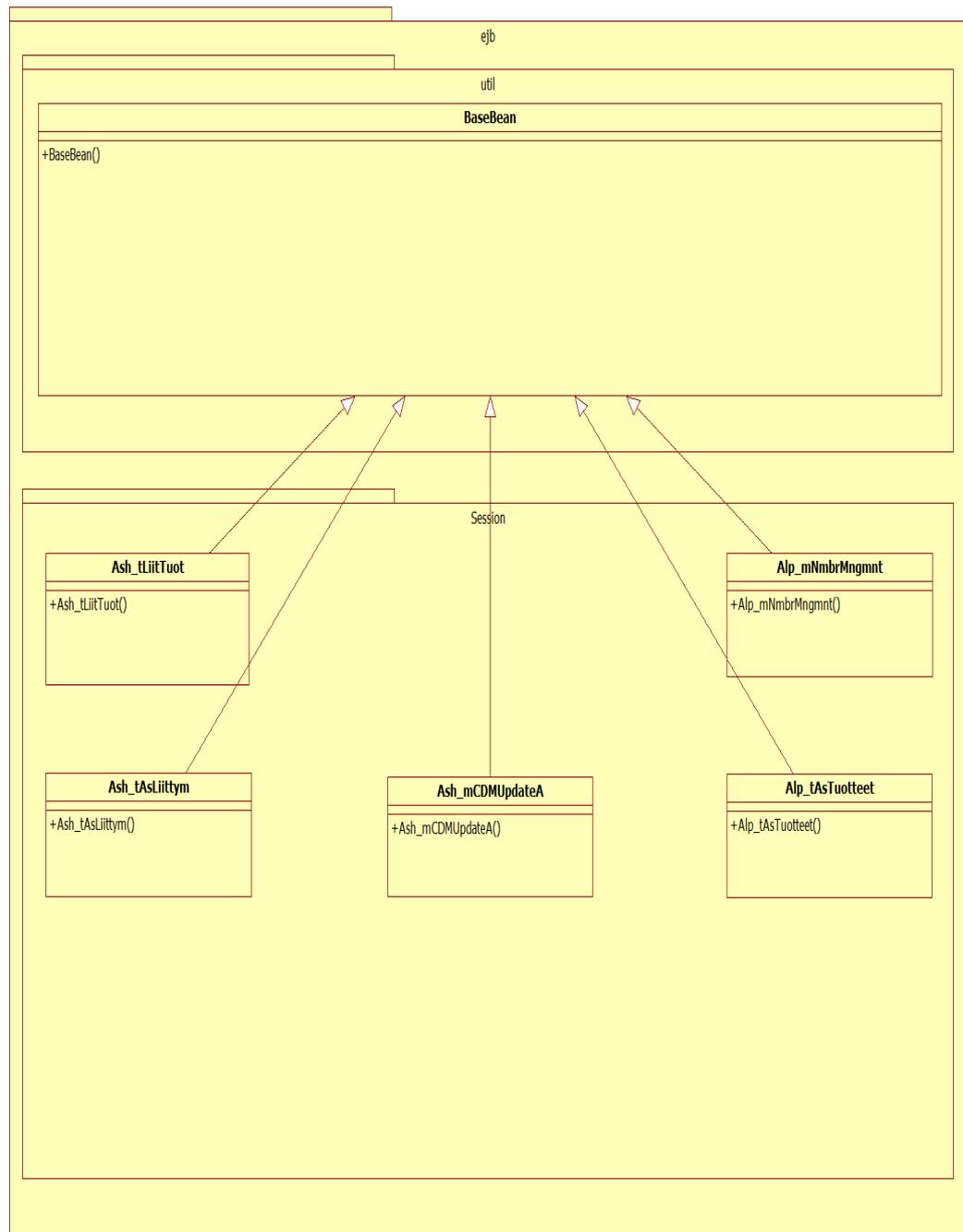
Stateless session bean Java-luokan ilmentymä luodaan silloin, kun EJB-containeri kutsuu tämän luokan newInstance-metodia. Seuraavaksi EJB-containeri kutsuu tämän luokan setSessionContext- ja ejbCreate-metodeja. Luokka on nyt valmis käytettäväksi mistä tahansa clientista, ja myös EJB-containeri voi kutsua luokan ejbTimeout-metodia, jos luokalle on asetettu ajastintoiminto. Kun EJB-containeri ei tarvitse enää luokan ilmentymää, niin se kutsuu luokan ejbRemove-metodia, joka poistaa luokan ilmentymän käytöstä. Stateless session bean Java-luokan toimintaperiaate on kuvattu kuviossa 20. (Enterprise JavaBeans™ Specification, Version 2.1 2003.)

5.2.9 Integraatio-sovelluksen luokkakaavio

Integraatio-sovelluksen luokkakaavio on kuvattu kuviossa 21. Kaikki palvelut periyttiin BaseBean-luokasta, johon toteutettiin XML-sanomien parserointi, Tuxedo-sanomien muunnokset XML-sanomiksi, XML-sanomien muunnokset Tuxedo-sanomiksi, JMS-sanomien käsittely ja virhekäsittelyyn liittyviä yleisfunktioita.



KUVIO 20. Stateless session bean Java-luokan toimintaperiaate



KUVIO 21. Integraatiosovelluksen package model

5.2.10 Integraatiosovelluksen BaseBean-luokan toiminnallisuus

Luokan attribuutteja käytetään JMS-sanomajonojen, debug-tulostusten ja poikkeustilanteiden käsittelyyn. `startXMLParse`-metodi aloittaa XML:n parseroinnin ja `stopXMLParse`-metodi lopettaa XML:n parseroinnin, näitä metodeja käytetään myös debug-tulostuksiin.

`createTagRow`-, `createStartTag`-, `printStartTag`- ja `printEndTag`-metodeja käytetään debug-tulostuksiin. `getRequestXMLEncoding`-, `getCDMRequestXMLEncoding`- ja `getReplyXMLEncoding`-metodeilla luetaan integraatio-sovelluksen käyttämät merkistöt, ja merkistöt välitetään parametreina integraatio-sovellukselle sen käynnistyksen yhteydessä.

`getReplyTimeOut`-metodilla luetaan integraatio-sovelluksen käyttämä timeout-arvo, eli miten kauan enintään sovellus odottaa vastausta laskutusjärjestelmän JMS-serveriltä. Jos vastausta ei saada timeout-arvoon mennessä, niin integraatio-sovellus palauttaa palvelun kutsujalle virheilmoituksen. Timeout-arvo välitetään parametrina integraatio-sovellukselle sen käynnistyksen yhteydessä. `getGenericTechnicalErrorCode`- ja `createErrorReply`-metodeja käytetään virhesanomien luontiin.

`addFieldToTuxedoMessage`- ja `addXMLTagToTuxedoMessage`-metodeilla laskutusjärjestelmän JMS-serveriltä vastaanotetut XML-sanomat muunnetaan Tuxedo-sanomiksi. `addTuxedoFieldToXML`-metodeilla ulkoisilta Tuxedo-domaineilta vastaanotetut Tuxedo-sanomat muunnetaan XML-sanomiksi. `copyTuxedoMessageHeaderFields`-metodilla kopioidaan jokaisen Tuxedo-sanomaan alkuun sanomakohtaiset otsikkokentät.

`startJMSConnection`-metodilla avataan yhteys ja `stopJMSConnection`-metodilla suljetaan yhteys integraatio-sovelluksen JMS-sanomajonoihin. `sendJMSRequest`-metodilla lähetetään sanoma integraatio-sovelluksen JMS-sanomajonoon ja `receiveJMSReply`-metodilla vastaanotetaan sanoma integraatio-sovelluksen JMS-sanomajonosta. BaseBean-luokan attribuutit ja metodit on kuvattu kuviossa 22.

BaseBean
<pre> -JMS_CONNECTION_FACTORY: String = "weblogic.jms.ConnectionFactory" -INDENT_STRING: String = " " -GENERIC_TECHNICAL_ERROR_CODE: int = 1200045 -jndi: InitialContext -queue_connection_factory: QueueConnectionFactory -queue_connection: QueueConnection -queue_session: QueueSession -send_queue: Queue -receive_queue: Queue -queue_sender: QueueSender -queue_receiver: QueueReceiver </pre>
<pre> <<create>>+BaseBean() +startXMLParse(xml_records: String, indent_level: int): Document +endXMLParse(doc: Document, indent_level: int) +createTagRow(tag_row: String, indent_level: int): String +createStartTag(tag_name: String, indent_level: int): String +printStartTag(tag_name: String, indent_level: int) +createEndTag(tag_name: String, indent_level: int): String +printEndTag(tag_name: String, indent_level: int) +getRequestXMLEncoding(): String +getCDMRequestXMLEncoding(): String +getReplyXMLEncoding(): String +getReplyTimeout(): long +createXMLHeader(): String +addFieldToTuxedoMessage(tuxedo_message: TypedFML32, field_id: int, field_type: int, field: String) +addXMLTagToTuxedoMessage(tuxedo_message: TypedFML32, field_id: int, field_type: int, element: Element, tag_name: String, indent_level: int): String +addXMLTagToTuxedoMessage(tuxedo_message: TypedCArray, element: Element, tag_name: String, indent_level: int): String +startJMSConnection(send_queue_name: String, receive_queue_name: String) +stopJMSConnection() +sendJMSRequest(doc: Document, encoding: String, service_name: String): String +receiveJMSReply(service_name: String, JMSMessageID: String): String +addTuxedoFieldToXML(tuxedo_message: TypedFML32, field_id: int, field_occurrence: int, doc: Document, parent_element: Element, tag_name: String) +addTuxedoFieldToXML(tuxedo_message: TypedCArray, doc: Document, parent_element: Element, tag_name: String) +getGenericTechnicalErrorCode(): int +createErrorReply(msg_type: int, error_code: int, error_description: String, errorReply: TypedFML32) +createErrorReply(error_code: int, error_description: String, errorReply: TypedCArray) +copyTuxedoMessageHeaderFields(msg_type: int, tuxedoRequest: TypedFML32, tuxedoReply: TypedFML32) </pre>

KUVIO 22. Integraatiosovelluksen BaseBean-luokan attribuutit ja metodit

5.2.11 Integraatiosovelluksen Ash_tLiitTuot-luokan toiminnallisuus

Ash_tLiitTuot-luokan attribuutit ja metodit on kuvattu kuviossa 23 ja luokan toiminnallisuus on kuvattu sekvenssi-kaaviona kuviossa 24. Luokan attribuutteja käytetään JMS-sanomajonojen käsittelyyn ja stateless session beanin sessionin tallentamiseen. Luokka toteuttaa SessionBean-rajapinnan setSessionContext-, ejbActivate-, ejbPassivate-, ejbRemove- ja ejbCreate-metodit. Luokkaa käytetään stateless session bean Java-luokkana, ja stateless session bean Java-luokan toimintaperiaate on kuvattu luvussa 5.2.8.

WebLogic Serverin Tuxedo Connector välittää ulkoisen järjestelmän Tuxedo-domainin kutsun tämän luokan service-metodille (1), jos domain kutsuu tämän luokan tarjoamaa palvelunimeä, kutsussa välitetään parametrina kutsujan lähettämä Tuxedo-sanoma. Luokan tarjoama palvelunimi konfiguroidaan WebLogic Serverin Tuxedo Connectorin konfiguraatiossa.

Luokka on periytetty integraatio-sovelluksen BaseBean-luokasta, joten luokasta voidaan kutsua esimerkiksi kantaluokan startJMSConnection-, sendJMSRequest-, receiveJMSReply- ja stopJMSConnection-metodeja. BaseBean-luokan toiminnallisuus on kuvattu luvussa 5.2.10.

Luokan service-metodi kutsuu kantaluokan startJMSConnection-metodia (2). Luokan service-metodi kutsuu luokan sendJMSRequest-metodia (3), joka muuntaa kutsujan lähettämän Tuxedo-sanoman XML-sanomaksi. Muunnoksen jälkeen service-metodi kutsuu kantaluokan sendJMSRequest-metodia (4), joka kirjoittaa XML-sanoman integraatio-sovelluksen JMS-sanomajonoon.

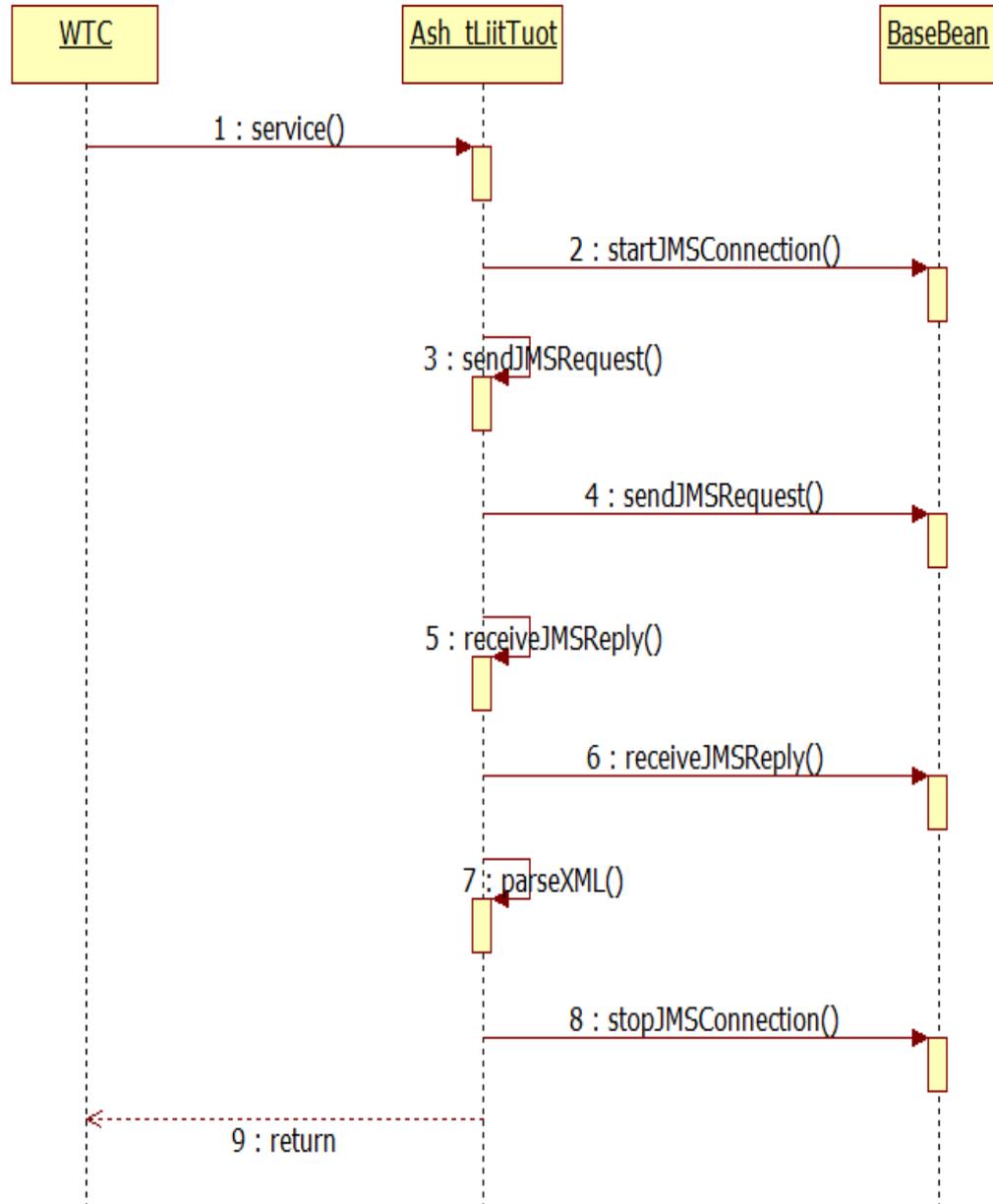
Luokan service-metodi kutsuu luokan receiveJMSReply-metodia (5), joka kutsuu edelleen kantaluokan receiveJMSReply-metodia (6), ja kantaluokan metodi jää lukemaan integraatio-sovelluksen JMS-sanomajona, sekä odottamaan vastausta laskutusjärjestelmän JMS-serveriltä.

Luokan service-metodi kutsuu luokan parseXML-metodia (7), joka muuntaa laskutusjärjestelmän JMS-serverin palauttaman XML-sanoman Tuxedo-sanomaksi. Muunnoksen jälkeen Tuxedo-sanoma palautuu service-metodille, joka kutsuu kantaluokan stopJMSConnection-metodia (8) ja lopuksi service-metodi palauttaa sanoman ulkoisen järjestelmän Tuxedo-domainille, eli palvelun kutsujalle (9).

Koska Ash_tAsLiittym-, Ash_mCDMUpdateA-, Alp_mNmbrMngmnt- ja Alp_tAsTuotteet-luokkien toiminnallisuus on vastaava kuin Ash_tLiitTuot-luokan toiminnallisuus, niin näiden luokkien toiminnallisuutta ei ole kuvattu erikseen tässä dokumentissa.

Ash_tLiitTuot
<pre>-serialVersionUID: long = 1261485843897497325L -ctx: SessionContext -JMS_SEND_QUEUE_NAME: String = System.getProperty("com.tsf.apm.integration.env", "test") + "_q.alr.atlas.internal.sami_ash_tliittuot.request" -JMS_RECEIVE_QUEUE_NAME: String = System.getProperty("com.tsf.apm.integration.env", "test") + "_q.alr.atlas.internal.sami_ash_tliittuot.reply" -JMSMessageID: String</pre>
<pre><<create>>+Ash_tLiitTuot() +setSessionContext(ctx: SessionContext) +ejbActivate() +ejbPassivate() +ejbRemove() +ejbCreate() +service(request_reply: TPServiceInformation): Reply +sendJMSRequest(tuxedoRequest: TPServiceInformation): TypedFML32 +receiveJMSReply(): TypedFML32 +parseXML(xml_records: String): TypedFML32</pre>

KUVIO 23. Integraatiosovelluksen Ash_tLiitTuot-luokan attribuutit ja metodit



KUVIO 24. Integraatiosovelluksen Ash_tLiitTuot-luokan toiminnallisuus sekvenssi-kaaviona

6 YHTEENVETO, JOHTOPÄÄTÖKSET JA KÄYTÄNNÖN KOKEMUKSET

Kaikki toteutusosuudessa tehdyt testit olivat suoraviivaisia toteuttaa, eikä niissä havaittu ongelmia. TAO CORBA -serverit vastasivat nopeasti clienttien kutsupyyntöihin, joten korvaavan sanomanvälitysohjelmiston suorituskyky todettiin myös hyväksi. TAO CORBA:n kuormantasauksella servereihin saatiin myös lisää skaalautuvuutta ja lisäksi asynkroniset palvelukutsut nopeuttivat clienttien toimintaa.

Projektissa uusittiin myös vanhat palvelimet, joiden käyttöjärjestelmänä oli HP-UX. Uusien palvelimien käyttöjärjestelmäksi valittiin Red Hat Enterprise Linux, ja eräohjelmien kääntämiseen käytettiin GNU C/C++ -kääntäjää. Uudessa ympäristössä tietyt ohjelmat eivät enää toimineet oikein, ja tarkempien tutkimusten jälkeen selvisi, että ohjelmissa oli ollut virheitä, mutta virheet eivät olleet tulleet esille vanhassa ympäristössä. Uusi C-kääntäjä oli myös tarkempi, ja ohjelmakoodia jouduttiin muuttamaan, jotta kääntäjä suostui kääntämään ne.

Projektissa korotettiin myös tietokannan versiota, Oracle-versiosta 8 siirryttiin versioon 9. Uuden ympäristön Pro*C-kääntäjä oli tarkempi ja ohjelmakoodia jouduttiin muuttamaan, jotta kääntäjä suostui kääntämään ne. Tiettyjä SQL-lauseita jouduttiin myös jonkin verran muuttamaan, koska ne toimivat liian hitaasti uudessa ympäristössä.

Käytännön kokemukset TAO CORBA -sanomanvälitysohjelmistosta ovat olleet hyviä, ja ohjelmiston suorituskyky havaittiin testeissä erinomaiseksi. WebLogic Server -integraatio on myös toiminut hyvin, ja serveri käynnistää automaattisesti tarvittavan määrän palveluja kuormituksen kasvaessa.

6.1 Laskutusjärjestelmän uusi arkkitehtuuri

Laskutusjärjestelmän sisällä sanomanvälitykseen käytetään TAO CORBA - sanomanvälitysohjelmistoa sekä käyttöliittymissä että eräajoissa. C# .NET - käyttöliittymien ja sanomanvälityskerroksen välissä on C-kielellä tehty sovituserros, johon on toteutettu sanomanvälitykseen tarvittava toiminnallisuus. PowerBuilder-käyttöliittymän ja sanomanvälityksen välissä on myös C-kielellä tehty sovituserros, johon on toteutettu sanomanvälitykseen tarvittava toiminnallisuus, ja lisäksi sovituserroksen toteutuksessa on käytetty PBNI eli PowerBuilder Native Interface -tekniikkaa.

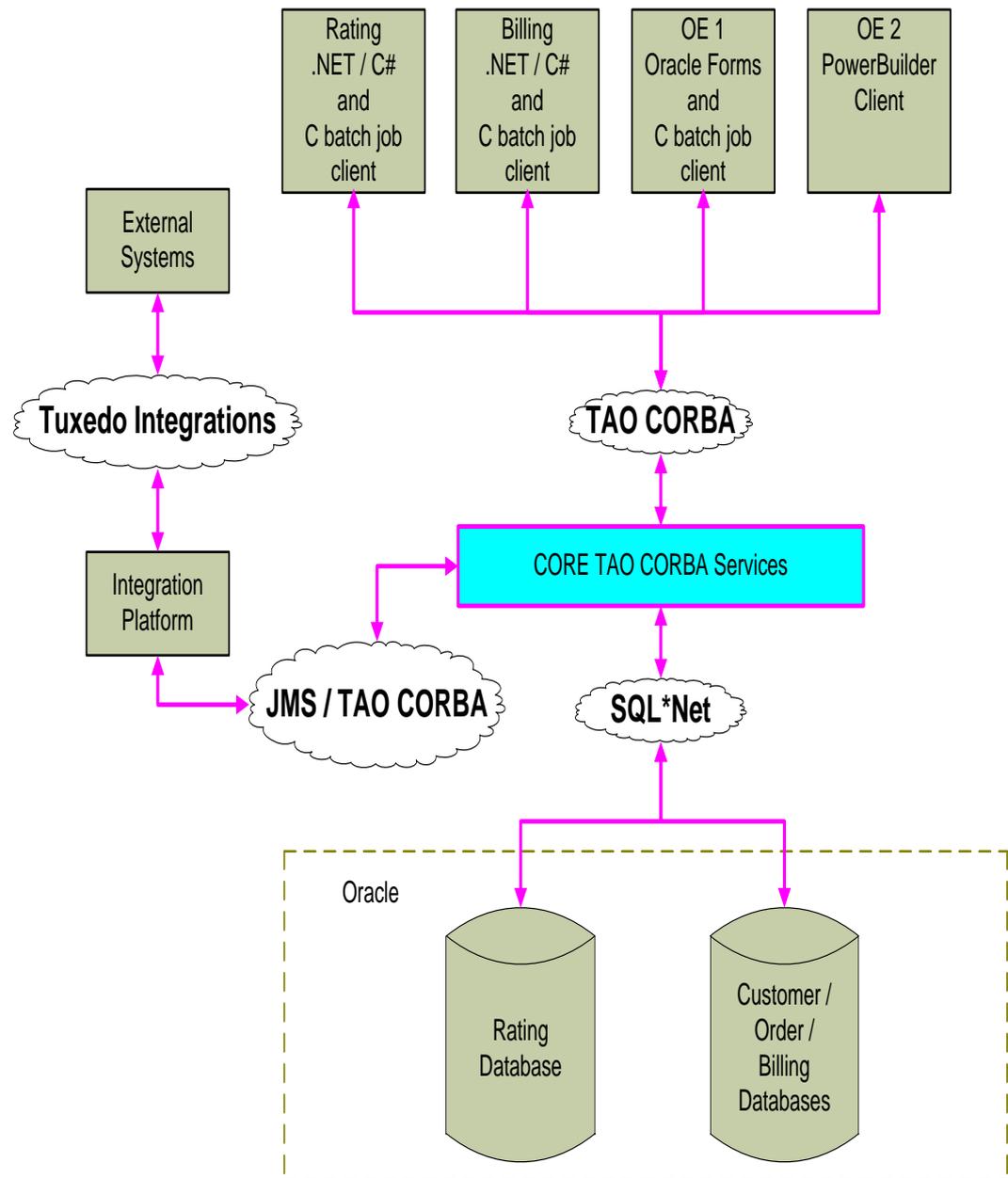
Integraatiot ulkoisiin järjestelmiin on toteutettu WebLogic Server -integraatioina. Laskutusjärjestelmän ja integraatio-ohjelmiston välissä on käytetty JMS eli Java Message Service -jonoja. Integraatio-ohjelmiston ja ulkoisten järjestelmien välissä on käytetty Tuxedo-sanomanvälitysohjelmistoa. Laskutusjärjestelmän palvelut piti tarjota ulkoisille järjestelmille edelleen Tuxedo-palveluina, koska Tuxedoa ei toistaiseksi korvattu ulkoisissa järjestelmissä.

Integraatio-ohjelmisto muuntaa ulkoisten järjestelmien Tuxedo-sanomat XML-sanomiksi ja kirjoittaa ne JMS-jonoon. Laskutusjärjestelmän JMS-serveri lukee pyyntösanoman jonosta, muuntaa sanoman TAO CORBA -sanomaksi, ja välittää sen laskutusjärjestelmän TAO CORBA -serverille, joka suorittaa varsinaisen palvelun. TAO CORBA -serveri palauttaa vastaussanoman JMS-serverille, joka muuntaa sen XML-sanomaksi ja kirjoittaa JMS-jonoon. Integraatio-ohjelmisto lukee vastaussanoman JMS-jonosta, muuntaa sen Tuxedo-sanomaksi ja palauttaa vastauksen ulkoisen järjestelmän clientille. Uusi laskutusjärjestelmän arkkitehtuuri on kuvattu sivulla 30 kuviossa 16, sivulla 31 kuviossa 17 ja sivulla 48 kuviossa 25.

6.2 TAO CORBA -sanomanvälitysohjelmiston tulevaisuus

TAO CORBA -sanomanvälitysohjelmistoa ylläpidetään ja kehitetään aktiivisesti OCI eli Object Computing Inc. yrityksessä, joka sijaitsee Yhdysvalloissa. Uusin ladattavissa oleva versio on 2.0a, ja tämä täyttää CORBA-versiolla 3.0 asetetut vaatimukset. (TAO 2.0a Downloads 2011.) Ohjelmistoa käytetään vaativissa sovelluksissa, ja sen käyttäjiä ovat esimerkiksi Yhdysvaltain armeija, Boeing, Siemens ja Motorola (OCI wins US Army FCS Award 2011; ACE, TAO, and CIAO Success Stories 2011).

TAO CORBA ja JacORB -sanomanvälitysohjelmat ovat yhteensopivia. JacORB on Javalla tehty Open Source CORBA -sanomanvälitysohjelmisto, ja OCI tarjoaa myös tälle ohjelmistolle kaupallista tukea. (JacORB 2011.)



KUVIO 25. Uusi laskutusjärjestelmän arkkitehtuuri

LÄHTEET

Douglas C. Schmidt. 2011.
Members of the ACE and TAO Development Team
[viitattu 7.5.2012]. Saatavissa:
<http://www.cs.wustl.edu/~schmidt/ACE-members.html>

Douglas C. Schmidt. 2011.
ACE, TAO, and CIAO Success Stories
[viitattu 7.5.2012]. Saatavissa:
<http://www.cs.wustl.edu/~schmidt/ACE-users.html>

Object Computing, Inc. 2011.
OCI's Support Model for ACE and TAO
[viitattu 7.5.2012]. Saatavissa:
<http://www.theaceorb.com/support/index.html>

Object Management Group, Inc. 2012.
OMG IDL: Details
[viitattu 7.5.2012]. Saatavissa:
http://www.omg.org/gettingstarted/omg_idl.htm

Object Management Group, Inc. 2012.
CORBA® BASICS
[viitattu 7.5.2012]. Saatavissa:
<http://www.omg.org/gettingstarted/corbafaq.htm>

Irfan Pyarali and Douglas C. Schmidt. 2012.
An Overview of the CORBA Portable Object Adapter
[viitattu 7.5.2012]. Saatavissa:
<http://www.dre.vanderbilt.edu/~schmidt/PDF/POA.pdf>

Ossama Othman, Carlos O’Ryan, and Douglas C. Schmidt. 2001.

The Design of an Adaptive CORBA Load Balancing Service

[viitattu 7.5.2012]. Saatavissa:

http://www.cs.wustl.edu/~schmidt/PDF/load_balancing2.pdf

Object Management Group, Inc. 2012.

ORB Basics

[viitattu 7.5.2012]. Saatavissa:

http://www.omg.org/gettingstarted/orb_basics.htm

Oracle. 2012.

Oracle WebLogic Server Overview

[viitattu 7.5.2012]. Saatavissa:

<http://www.oracle.com/technetwork/middleware/weblogic/overview/index.html>

Oracle. 2008.

Introducing Oracle Tuxedo ATMI

[viitattu 7.5.2012]. Saatavissa:

http://docs.oracle.com/cd/E13161_01/tuxedo/docs10gr3/int/intarch.html

Sun Microsystems. 2003.

Enterprise JavaBeans™ Specification, Version 2.1

[viitattu 7.5.2012]. Saatavissa:

http://www.cs.helsinki.fi/u/przybils/courses/CBD06/papers/ejb-2_1-fr-spec.pdf

Object Computing Inc. 2011.

TAO 2.0a Downloads

[viitattu 7.5.2012]. Saatavissa:

<http://www.theaceorb.com/downloads/2.0a/index.html>

Object Computing Inc. 2011.

OCI wins US Army FCS Award

[viitattu 7.5.2012]. Saatavissa:

<http://www.ociweb.com/news/OCI-Wins-US-Army-FCS-Award>

Object Computing Inc. 2011.

JacORB

[viitattu 7.5.2012]. Saatavissa:

<http://www.ociweb.com/products/jacorb>