CORBA

Avoimien hajautettujen järjestelmien tutkimusseminaari
Manu Karjalainen
Tietojenkäsittelytieteen laitos
11.2.2000

1. Johdanto

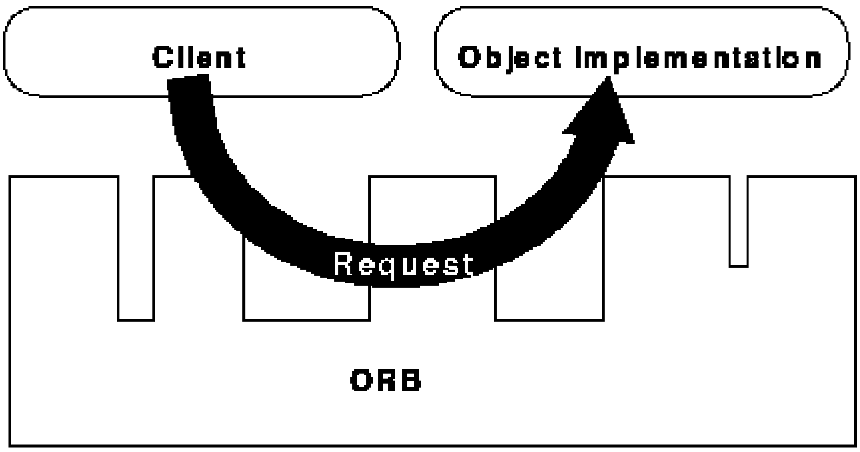
CORBA on määritelmä ja standardi erilaisten hajautettujen oliopohjaisten ohjelmistojen yhteen liittämiseen. CORBA standardin perusidea on se, että ohjelmat tai komponentit voivat kommunikoida keskenään tietyn CORBA-toteutuksen sisällä vaikka ne olisi kirjoitettu eri ohjelmointikielillä tai toimisivat eri laitteistoissa tai käyttöjärjestelmissä. CORBA hoitaa ns. "likaisen työn" komponenttien välisessä kommunikaatiossa. CORBA itsessään ei ole ohjelmointikieli vaan eräänlainen liima erilaisten komponenttien välillä.

CORBA perustuu asiakas-palvelin arkkitehtuuriin, jossa palvelin oliot tarjoavat palveluita asiakasolioille. Oliot voivat olla kokonaisia ohjelmia tai erillisiä ohjelman osia. CORBA:n tarjoamiin palveluihin kuuluu mm. komponenttien paikantaminen ja aktivointi, virheidenkäsittely, parametrivälitys sekä tiedon formaatin muunnos eri arkkitehtuurien välillä.

OMG (Object Management Group), jonka tarkoituksena on edistää oliotekniikoita hajautettujen järjestelmien käyttöön, julkaisi ensimmäisen CORBA 1.0 spesifikaation vuonna 1989. Sitä seurasi vuonna 1991 CORBA 1.1. Se oli ensimmäinen askel kohti objektien välistä vuorovaikutusta mutta ei ollut mitenkään täydellinen spesifikaatio. CORBA 2.0 määritteli protokollan joka laajensi määrittelyn koskemaan kommunikaatiota myös eri valmistajien CORBA-toteutuksien välillä . Tätä protokollaa kutsutaan myös nimellä Internet Inter-ORB protocol (IIOP). Nykyään on määritelty jo CORBA 3.x standardi, jossa on pyritty parantamaan mm. CORBA:n tietoturvaominaisuuksia. CORBA:a on käytetty eri ohjelmistoissa vuodesta 1992 vaihtelevalla menestyksellä[1].

2. CORBA toimintakuvaus

Kuvassa 1 näkyy palvelupyyntö asiakkaalta (Client) objektin toteutukselle Object Implementation) . Asiakas on ohjelma tai ohjelman osa, joka haluaa suorittaa operaation johonkin objektiin. Objektin toteutus on koodi ja data, joka toteuttaa objektin. ORB on vastuussa kakista tarvittavista toimenpiteistä, jotka liittyvät objektin toteutuksen löytämiseen, valmistaa kohdeobjektin pyyntöön ja kommunikointiin datan kanssa eli pyynnön tekemiseen. Lisäksi ORB hoitaa operaation tuloksen palautuksen takaisin asiakkaalle. Rajapinta jonka asiakas näkee on täysin riippumaton kohdeobjektin sijainnista tai siinä käytetystä ohjelmointikielestä[3].



Kuva1 : Palvelupyynnön kulku.

2.1 Client

CORBA toteutuksessa asiakas on ohjelma tai prosessi joka käyttää jonkin objektin tarjoamia palveluja ja palvelin on ohjelma tai komponentti joka tarjoaa palveluja. Asiakkaalla on viittaus (Object Reference) johonkin objektiin, jonka avulla se voi kustua objektin tarjoamia palveluja. Objektin viite on informaatio, jolla ORB voi tunnistaa halutun objektin. Asiakas voi olla asiakas monelle eri objektille sekä objekti voi itse olla asiakas jollekin muulle objektille. Asiakas tietää objektin loogisen rakenteen ja liitynnän muttei koskaan kohdeobjektin toteutuksen yksityiskohtia. Tämä takaa toteutuksen vaihdettavuuden liitynnän takana.

2.3 Object Implementation

Object implementation eli objektin toteutus sisältää objektin todellisen tilan ja käyttäytymisen. Toteutus voi rakentua monin eri tavoin. Omien metodien ja operaatioiden lisäksi toteutus voi määritellä proseduureja jotka aktivoivat ja deaktivoivat muita objekteja.

3. ORB

3.1 ORB:N yleiskuvaus

ORB on CORBA:n tärkein osa. OMG:n määritelmän mukaan ORB tuottaa mekanismin, joilla oliot läpinäkyvästi tekevät palvelupyyntöjä ja vastaanottavat tuloksia. ORB mahdollistaa sovellusten keskenään toimivuuden hajautetuissa ympäristöissä ja kytkee yhteen useampia oliojärjestelmiä. ORB helpottaa hajautettujen sovellusten ohjelmointia kätkemällä tietoliikenteeseen liittyvät rutiinitoimet, niin että palvelun kutsu näyttää asiakkaalle paikalliselta funktio tai metodikutsulta. Palvelupyynnöt eivät välity suoraan asiakkaalta oliolle, vaan palvelupyynnöt hoitaa aina ORB. Jokainen CORBA-kutsu välitetään ORB:lle. Kutsun aikana ORB ensin etsii palvelun toteutuksen, muuntaa ja siirtää parametrit, aktivoi tarvittaessa toteutuksen, siirtää kontrollin palveluobjektille ja palauttaa saadut tulokset takaisin asiakkaalle. Kutsun muoto on sama riippumatta siitä onko kohdeolio paikallinen vai ulkopuolinen. Jos kohdeolio on ulkopuolinen, niin kutsu välittyy asiakkaan ORB:ltä kohdeolion toteutuksen ORB:lle[2].

Kun komponentti haluaa käyttää jonkin toisen komponentin palvelua, sillä täytyy olla kyseisen komponentin viite. Kun viite on hallussa, komponentti voi käyttää halutun komponentin metodeja ja saada haluttuja tuloksia. Yleensä metodeissa käytetään syöttö ja tulostusparametrejä. ORB:n yksi vastuualue on vastaanottaa syöteparametrit kutsuvalta komponentilta ja muuttaa parametrit "on-the-wire" formaattiin eli muotoon, jossa ne voivat matkata verkon yli vastaanottavalle objektille. Tästä käytetään termiä marshalling. Vastaanottajan päässä ORB muuttaa parametrit on-the-wire formaatista sellaiseen muotoon jota kutsuttava komponentti ymmärtää. Tästä käytetään termiä unmarshalling. Tämän jälkeen kutsuttava komponentti lähettää paluuarvon jonka ORB muuttaa taas on-the-wire formaattiin lähetettäessä ja kutsujan päässä ORB muuttaa sen alkuperäisen kutsujan ymmärtämään muotoon[2].

ORB kätkee seuraavat asiat:

Palveluobjektin sijainti: Asiakkaan tarvitsee vain antaa ORB:lle haluamansa palvelun nimi tai ominaisuudet, eikä sen tarvitse tietää missä palveluobjekti sijaitsee. Se voi sijaita eri koneessa jossain päin verkkoa, samassa koneessa eri prosessissa tai samassa prosessissa.

Palveluobjektin toteutus:
Asiakas ei tiedä millä ohjelmointikielellä palvelu on toteutettu tai missä käyttöjärjestelmässä se toimii.

Palveluobjektin tila: Asiakas ei tiedä onko palvelu aktiivisena valmiina hyväksymään pyyntöjä, kun se kutsuu palvelua. ORB käynnistää ja aktivoi palvelun tarvittaessa.

Kommunikaatiomenetelmän: Asiakas ei tiedä millä protokollalla ORB kutsuu palvelua. (paikallinen funktiokutsu, TCP/IP jne.)

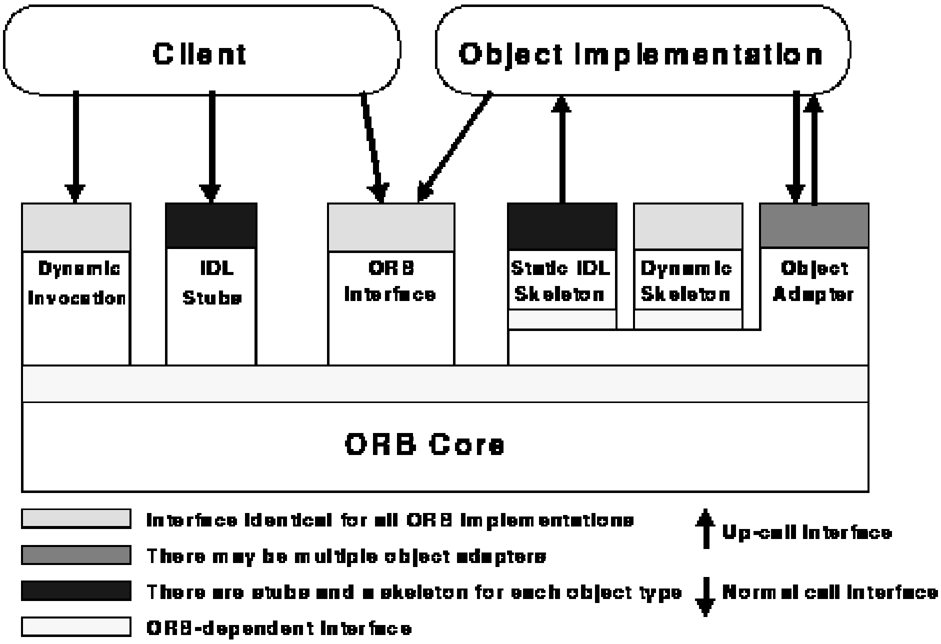
IDL

OMG IDL (Interface Definintion Language) on kommunikaatiorajapintojen kuvaamiseen tarkoitettu kieli joka muistuttaa hieman C:tä tai C++:aa. Se sisältää rajapinnan funktioiden ja tietotyyppien esittelyn. IDL:n avulla kohdeobjekti kertoo asiakkailleen mitä operaatioita on mahdollista suorittaa ja miten ne suoritetaan. IDL määrittelyä voisi verrata vaikkapa C++:n header tiedostoon. IDL määrittelyjen avulla on mahdollista mapata eli yhdistää CORBA objektit käytettävissä olevaan ohjelmointikieleen ja systeemiin.

3.2 ORB:N rajapintojen kuvaus

Kuvassa 2 on tyypillisen ORB:n rajapintojen rakenne. ORB:n rajapinnat on kuvattu laatikoilla ja nuolet osoittavat kutsutaanko ORB:tä vai suorittaako ORB kutsun rajapinnan yli.

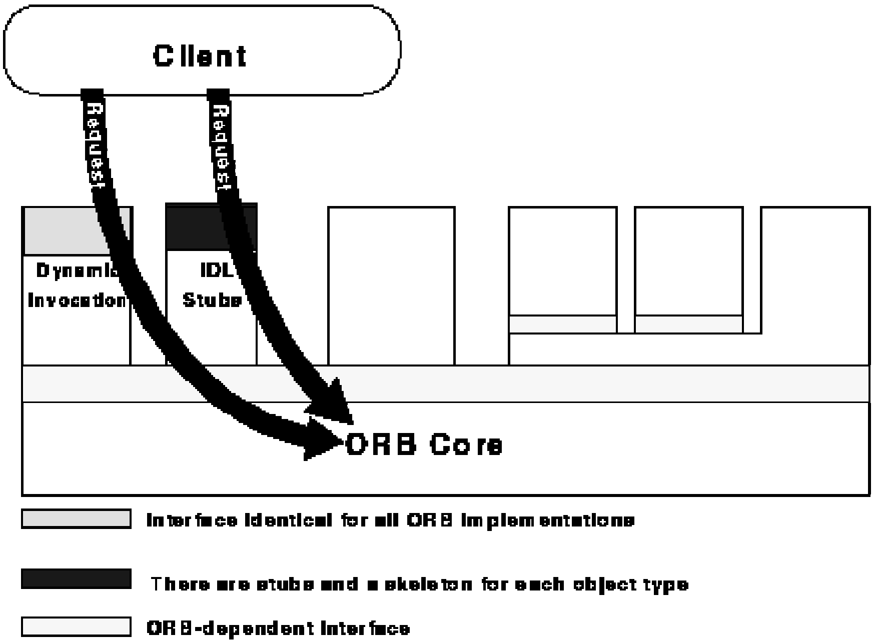
Asiakkaan puolen rajapintoja ovat Dynamic Invocation interface (DII), OMG IDL stub ja ORB interface. Vastaavasti palvelimen puolen rajapintoja ovat Static IDL Skeleton, Dynamic skeleton sekä Object Adapter[3].



Kuva 2: ORB:n rakenne

3.3 Palvelupyyntöjen kuvaus

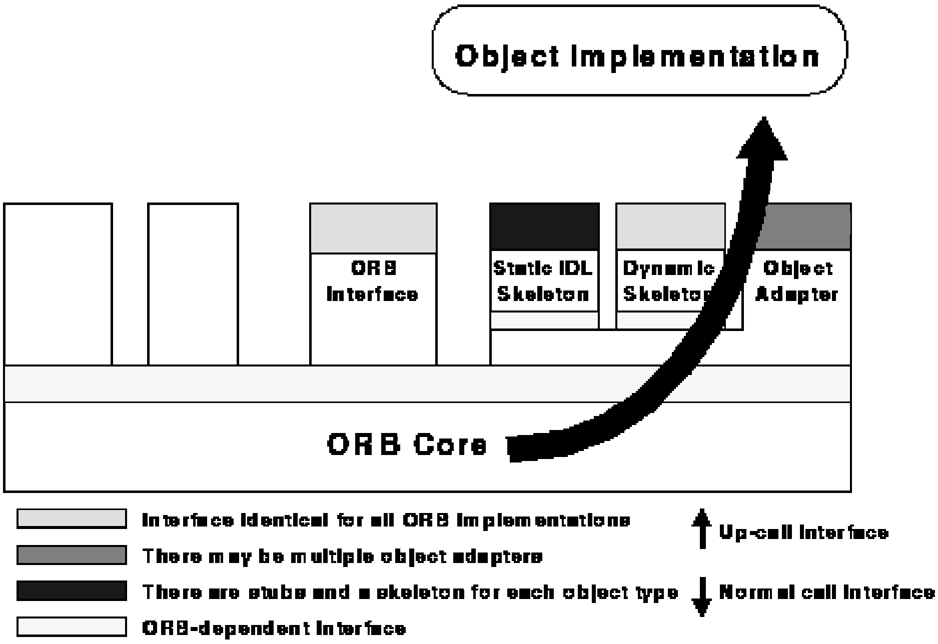
Kuvassa 3 näkyy kuinka asiakas tekee palvelupyynnön. Asiakkaalla on kaksi vaihtoehtoa määritellä käytettävä rajapinta. Rajapinnan voi määritellä staattisesti IDL:n avulla. Tätä menetelmää kutsutaan myös termillä SII (Static Invocation Interface). Toinen tapa on muodostaa palvelupyyntö dynaamisesti DII:n (Dynamic Invocation Interface) avulla. Asiakas voi myös suoraan kommunikoida ORB:n kanssa eräiden funktioiden avulla . Palvelin ei ”näe” kummalla menetelmällä (SII vai DII) asiakas on sitä kutsunut[3].



Kuva 3: Palvelupyynnön lähetys.

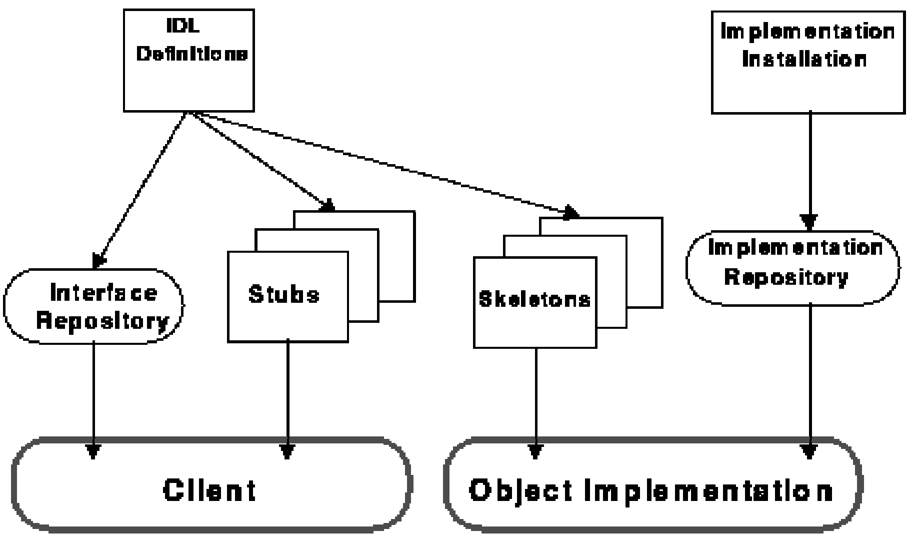
3.4 Palvelupyyntöjen vastaanoton kuvaus

Kuvassa 4 näkyy kuinka palvelimen objektin toteutus saa palvelupyynnön. ORB paikallistaa halutun toteutuksen koodin, jonka jälkeen se kuljettaa saamansa parametrit ja siirtää ne kohdeobjektille staattisen IDL Skeleton:in tai dynaamisen skeleton:in kautta. Kohdeobjekti voi käyttää ORB:n palveluita Object Adapterin kautta samalla kun se suorittaa palvelupyyntöä[2].



Kuva 4: Palvelupyynnön vastaanotto

Kuvassa 5 näkyy kuinka rajapinnat ja toteutuksen informaatio on asiakkaan objektin toteutuksen saatavilla. Rajapinta on määritelty OMG IDL:ssä tai/ja Interface Repositoryssä. Määritelmän mukaan generoidaan stubit ja skeletonit. Objektin toteutuksen tiedot saadaan installointivaiheessa ja se talletetaan Implementation repositoryyn[3].



Kuva 5: Interface ja Implementation Repository

3.5 ORB rajapintojen rakenne[3]

3.5.1 ORB Interface

ORB rajapinta on sama kaikille ORB toteutuksille ja sen palvelut liittyvät suoraan ORB ytimeen. ORB Rajapinnan palvelut eivät riipu palveluobjektin rajapinnoista tai objektisovittimesta (Object Adapter). Kuitenkin melkein kaikki ORB:n toiminnot ”ovat tarjolla” vain stubs:ien, skeleton:ien, DII:n tai objektisovittimen kautta, joten kaikille objekteille on vain muutama yhteinen operaatio. Näitä operaatioita käytetään sekä asiakkaan että palvelimen puolilla.

3.5.2 Dynamic Invocation Interface

DII on palveluobjektin rajapinnasta riippumaton ja sen avulla asiakas voi kutsua objektin palveluita, vaikkei niitä olisi tiedetty asiakkaan käännösvaiheessa. Asiakas kertoo DII:lle kutsuttavan palveluobjektin, funktion sekä käytettävät parametrit ja niiden tyypit. Nämä tiedot saadaan tavallisesti CORBA:n Interface Repository:sta.

Kutsu voidaan toteuttaa kolmella eri tavalla:

Synkroninen: Asiakas kutsuu palvelua ja jää odottamaan vastausta. Tätä tapaa käytetään myös SII:n IDL stub:eissa ja se on tavallisin tapa.

Viivästetty synkroninen: Asiakas kutsuu palvelua, palaa heti ja noutaa tulokset myöhemmin. Kätevä tapa, jos asiakas haluaa kutsua useita eri palveluita, joiden suoritus kestää kauan.

Yksisuuntainen: Asiakas lähettää vain pyynnön mutta ei odota vastausta. Toimii myös SII:llä.

Dynamic Skeleton Interface:
DSI on palvelinpuolen vastine DII:lle. DSI mahdollistaa palvelun kirjoittamisen, vaikka sen toteuttaman objektin tyyppi ei olisi tiedossa käännösaikana. DSI:n funktioilla kerrotaan ORB:lle oalvelun toteuttama funktio ja sen parametrit.

DII on joustavampi kuin SII, koska DII sallii useita kutsutapoja, eikä palvelua tarvitse tuntea käännösvaiheessa. Toisaalta se on myös monimutkaisempi ja hitaampi.

3.5.3 Static Invocation Interface

OMG IDL-kääntäjä tuottaa IDL-kuvauksesta asiakkaan päähän ”stubin” ja palvelimen päähän ”skeletonin”.
Stub on funktio, joka kutsuu palveluobjektia asiakkaan puolesta ORB:n avulla. Se näyttää asiakkaalle mahdollisimman paljon samalta, kuin IDL-kuvauksen palvelufunktio. Stub muuntaa hoitaa parametrien muunnoksen ohjelmointikielen muodosta on-the wire muotoon.
Skeleton on funktio, jonka avulla kutsu siirtyy ORB:ltä palvelun toteuttavalle objektille. Skeleton muuttaa parametrit on-the-wire muodosta palvelinobjektin ymmärtämään muotoon ja kutsuu palvelun toteutusfunktiota. Lopulta toteutusfunktio palauttaa tulokset. Skeleton muuntaa ne verkkosiirtoon sopiviksi ja paluttaa ne ORB:n välityksellä. Asiakkaan päässä stub palauttaa tulokset asiakkaalle. Kutsua stubin ja skeletonin avulla sanotaan staattiseksi kutsuksi ja niiden muodostamaa rajapintaa sanotaan Static Invocation Interface:ksi.

3.5.4 Object Adapter

ORB ei aina voi etukäteen tuntea kaikkia mahdollisia objektityyppejä ja niiden rajapintoja. Silti ORB:n on pystyttävä aktivoimaan palveluobjekti tarvittaessa tai löydettävä oikeat objektit ja kutsua niiden metodeja. Vastaavasti objekti tarvitsee ORB:ltä tiettyjä palveluita. OA toimii ORB:n ja objektin välissä. Jos ORB ydin

toteuttaa objektin kaipaaman palvelun niin OA vain kutsuu ORB ydintä. Muutoin sen täytyy itse toteuttaa objektin haluama palvelu.
CORBA määrittelee joukon perusadaptereita. Tavallisimmat ovat Basic Object Adapter sekä Library Object Adapter ja niiden tulisi olla saatavilla jokaisessa ORB toteutuksessa.
Tyypillisiä adapterien tarjoamia palveluita ovat esim.

- Objektien viittausten luonti ja käännös
- Toteutuksen aktivointi ja deaktivointi
- Yksittäisten objektien aktivointi ja deaktivointi
- Metodien suoritus skeletonien kautta

3.5.5 Interface Repository

Interface Repository on eräänlainen ”rajapinta varasto” johon on talletettu palvelujen rajapintakuvaukset. ORB voi käyttää Interface Repository:n tietoja suorittaessaan kyselyjä. Niiden avulla ohjelman on mahdollista päättää, mitkä ovat objektin sallittuja operaatioita ja suorittamaan niitä sellaisten objektien kanssa joiden rajapinta ei käännösvaiheessa ollut tiedossa.
Lisäksi IR:ssä on yleensä varastoituna lisätietoa ORB objektien rajapinnoista esim. virheenjäljitys tietoja, stub ja skeleton kirjastoja ja rutiineja joilla voi formatoida tai selailla tietyn tyyppisiä objekteja.

3.5.6 Implementation Repository

Implementation Repository on eräänlainen ”toteutusvarasto”, joka sisältää tietoa millä ORB paikallistaa ja aktivoi objektien toteutuksia. Tavallisesti toteutusten installointi sekä toteutusten aktivointiin ja suoritukseen liittyvät toimet tehdään Implementation Repository:ssä olevien operaatioiden mukaan.
Lisäksi Implementation Repository:ssä on lisätietoja objektien toteutuksista esim. virheenjäljitys tietoja, resurssien allokointia, turvallisuusjuttuja.

4. CORBA palveluita

CORBA on osa OMA (Object Management Architecture), joka tarjoaa lukuisia yleisiä palveluita, jotka ovat käyttökelpoisia sovelluksille. Standardi ei määrittele mitä palveluita CORBA-sovelluksen tulisi tarjota. Palvelut riippuvat eri CORBA tuotteiden tarjoajien mukaan.

Erilaisia palveluita[2]:

- Concurrency control service: Tarjoaa rajapinnan samanaikaisuuden hallintaa varten

- Event service: Tarjoaa mekanismin jonka avulla CORBA objektit voivat lähettää ja vastaanottaa tapahtumia.

- Licensing service: Palvelun tarjoaja voi määritellä säännöt joiden avulla se voi kontrolloida palelun käyttöä.

- Life cycle service: Mahdollistaa CORBA objektien luonnin, tuhoamisen, kopioimisen ja siirtämisen

- Naming service: Mahdollistaa CORBA objektien rekisteröitymisen sekä paikantamisen nimen avulla

- Object trader service: Mahdollistaa sen, että muut objektit voivat paikallistaa CORBA objekteja.

Lisäksi löytyy paljon muita palveluita esim.

- Persistent object service:, Property service, Query service, Relationship service, Security service, Time service, Transaction service.

5. CORBA yhteenveto

Yleisesti CORBA täyttää hyvin hajautetuille järjestelmille asetetut vaatimukset. Laajennettavuus on CORBA:n perusominaisuus. Hyvä standardointi ja oliopohjaisuus takaavat laajennettavuuden. Oliopohjaisuuden tuoma tiedon kätkentä ja ORB:n tuoma olioiden kätkentä toteuttavat aidosti yhdenjärjestelmän näkymän. Kuitenkaan CORBA ei ole vielä täydellinen asiakas-palvelin ympäristö vaan siinä on puutteelliset turvallisuus- ja aikapalvelut. Turvallisuuspalvelut tukevat olioiden tunnistamista ja käyttöoikeuskontrollia ja aikapalvelut tukevat ajan tahdistamista. Näiden palveluiden toteuttaminen on tekeillä. Lisäksi CORBA on aika monimutkainen ja se vaatii ohjelmiston toteuttajalta paljon tietämystä.

Lähdeluettelo

[1] Object Management Group:What is CORBA
<http://www.omg.org/about/wicorba.htm>

[2] Rosenberg Jeremy, Teach yourself CORBA in 14 days

[3] <http://www.infosys.tuwien.ac.at/Research/Corba/OMG>