KONSEP PEMROGRAMAN JARINGAN DENGAN MEMANFAATKAN MIDDLEWARE ORB (OBJECT REQUEST BROKER)

Maman Somantri
Jurusan Teknik elektro undip
Jl. Prof. Sudharto, Tembalang – Semarang
Email: mmsomantri@elektro.ft.undip.ac.id
http://www.elektro.undip.ac.id/mmsomantri

Abstrak
Dalam pemrograman client server tingkat lanjut, adalah memungkinkan untuk membangun sebuah aplikasi dengan dasar platform pemrograman yang berbeda-beda. Dalam pemrograman jaringan biasa / konvesional, maka tidak akan mampu untuk mengkoneksikan dua atau lebih platform yang berbeda. Misalnya ada dua buah server aplikasi yang dibangun oleh masing-masing Java dan C++, sedangkan client-nya dibangun oleh aplikasi delphi. Untuk membangun aplikasi itu maka dibutuhkan sebuah lapisan yang bisa menghubungkan platform pemrograman yang berbeda, lapisan yang dimaksudkan adalah diistilahkan sebagai 'middleware'. Middleware pada tataran implementasi merupakan sebuah paket program instan yang dipakai pada suatu platform permograman tertentu, sedangkan pada tataran konsep, middleware merupakan sebuah lapisan untuk lalulintas penghubung komunikasi antar objek dari sistem yang berbeda. Ada beberapa jenis middleware, seperti ORB (Object Request Broker), RMI (Remote Method Invocation), dan MOM (Message Oriented Middleware). Dalam tulisan ini akan dibahas tentang tinjauan konseptual untuk membangun sebuah aplikasi dengan memanfaatkan pemrograman jaringan dengan middleware. Middleware yang akan digunakan adalah ORB (Object Request Broker).
Kata kunci : Middleware, ORB, CORBA

I. Pendahuluan

Untuk membangun sebuah aplikasi client server *multi platform programming* salah satu cara adalah dengan memanfaatkan middleware. Middleware bisa dijelaskan pada 2 tataran, yaitu tataran konsep/paradigma dan tataran aplikasi pemrograman. Pada tataran konsep, middleware digunakan sebagai jembatan atau penghubung dua aplikasi atau lebih yang memiliki perbedaan, middleware kadang disebut sebagai *plumbing* karena middleware digunakan untuk menghubungkan 2 bagian dari sebuah aplikasi dan digunakan untuk melewatkan data diantara mereka. [1]
Pada tataran implementasi, middleware adalah sebuah paket program instan yang digunakan untuk menghubungkan dua program yang berbeda platforma atau produk/vendor.
Beberapa jenis middleware yang bisa digunakan untuk pemrograman adalah ORB, RPC, RMI, DCE, dan MOM. Produk Middleware biasanya di pakai oleh sebuah arsitektur atau framework pemrograman. ORB (Object Request Broker) dimiliki oleh arsitektur atau framework pemrograman CORBA (Common Object Request Broker Architectur) dan JAVA, RMI (Remothe Method Invocation) dimiliki oleh teknologi JAVA.

Pada tulisan ini akan dijelaskan konsep dan implementasi pemrograman middleware ORB pada framework CORBA, dengan kasus client server berbeda platform pemrograman.
Tulisan ini akan dibagi menjadi beberapa bagian. Bagian berikutnya adalah bagian II akan membahas tentang middleware-middleware yang bisa membangun aplikasi bisnis (EAI). Pada bagian ini juga akan dibahasa CORBA-ORB serta arsitektur frameworknya. Pembahasan tentang konsep dan implementasi pemrograman dengan middleware ini akan dibahas pada bagian III dan pada bagian IV akan dibahas tentang tinjauan kritis penggunaan middleware dalam membangun aplikasi client server.

II. Midleware

2.1 Middleware dan Enterprise Application Integration

Middleware adalah sebuah aplikasi yang secara logic berada diantara lapisan aplikasi (application layer) dan lapisan data dari sebuah arsitektur layer-layer TCP/IP [1]. Middleware bisa juga disebut protokol. Protokol komunikasi middleware mendukung layanan komunikasi aras tinggi.
Biasanya program middleware menyediakan layanan pesan (*messaging services*) sehingga aplikasi-aplikasi yang berbeda-beda itu dapat berkomunikasi. Sistem middleware mengikat aplikasi-aplikasi yang terpisah. Penggunaannya dalam aplikasi bisnis dikenal sebagai Enterprise Application Integration (EAI).
Middleware EAI membuat penghubung antara aplikasi-aplikasi dalam berbagai cara, namun secara umum cara ini diistilahkan sebagai transformasi dan *routing* data dan mengatur aliran proses bisnis. Ada implikasi dalam hal ini bahwa aplikasi-aplikasi itu berada dalam sebuah dunia heterogenitas – perbedaan platform operasi, pemisahan model data dan penyimpan data, heterogenitas jaringan dan protokol komunikasi.

2.2 Middleware ORB dalam framework CORBA

CORBA (*Common Object Request Broker Architecture*)[2] adalah sebuah standar sistem terdistribusi yang dikembangkan oleh OMG (*Object Management Group*), yaitu sebuah konsorsium yang

terdiri dari lebih dari 800 perusahaan untuk membantu dalam pemrograman objek-objek terdistribusi. CORBA adalah sebuah cara bagi objek-objek untuk melakukan interoperasi lintas jaringan. Sejak spesifikasi CORBA versi 1.2 diperkenalkan pada tahun 1991, CORBA memberikan sebuah mekanisme standar bagi komunikasi antar objek lintas jaringan, kemudian spesifikasi CORBA tersebut berkembang dengan diperkenalkannya CORBA 2.0 di tahun 1994 dan CORBA 3.0 yang di-*release* tahun 2000. Hal yang penting untuk dicatat bahwa CORBA hanya sebuah spesifikasi untuk membuat dan menggunakan objek-objek terdistribusi, CORBA bukan sebuah produk atau bahasa pemrograman. Vendor-vendor yang ingin membuat produk-produk yang mengikuti spesifikasi CORBA dapat bebas untuk melakukannya. Tetapi yang perlu ditekankan/diyakinkan adalah bahwa vendor-vendor tersebut mengikuti spesifikasi CORBA secara persis sama. Hal ini agar semua produk yang dihasilkan vendor-vendor tersebut memiliki keselarasan dengan CORBA (CORBA *compliant*), sehingga satu sama lain dapat berinteraksi. Hal lain yang perlu diingat bahwa CORBA independen terhadap bahasa pemrograman selama bahasa-bahasa pemrograman tersebut memiliki pemeta (*mapping*) dari bahasa definisi interface dalam CORBA.

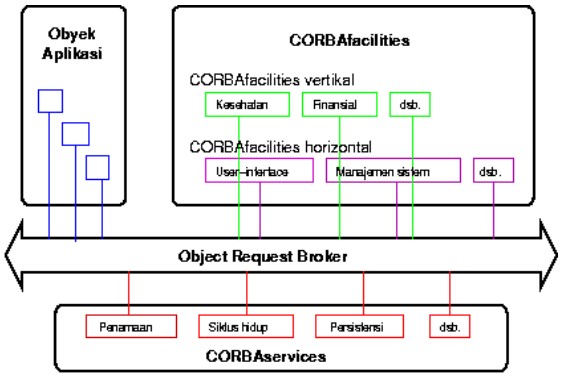
CORBA merupakan sebuah spesifikasi *middleware* yang ideal untuk mendukung dan mengaplikasikan sistem komputer terdistribusi. Arsitektur CORBA berbasis pada model objek. Model ini berasal dari abstraksi inti model objek yang didefinisikan oleh OMG dalam sebuah petunjuk OMA (*Object Management Architecture*), yang dapat ditemukan dalam [2]. Beberapa hal penting yang perlu dicatat bahwa CORBA berbeda dengan pemrograman objek serupa adalah:

- Objek-objek CORBA dapat berjalan dalam berbagai platform.
- Objek-objek CORBA ditempatkan di manapun dalam jaringan.
- Objek-objek CORBA dapat ditulis dalam beberapa bahasa pemrograman yang memiliki pemeta IDL (*IDL mapping*).

Hal yang juga dimiliki CORBA adalah CORBA bersifat *open*, maksudnya bahwa CORBA bisa dipakai oleh setiap orang yang ingin menggunakan standarisasi CORBA ini. Sehingga akan muncul perbedaan-perbedaan dalam menggunakannya, seperti perbedaan platform ataupun bahasa pemrograman. Tetapi hal ini justru menjadi kelebihan CORBA bahwa CORBA mampu mengkomunikasikan sistem yang memiliki perbedaan-perbedaan tersebut.

CORBA merupakan sebuah arsitektur yang menyediakan sebuah framework *crossplatform* untuk membangun dan mengembangkan sistem objek terdistribusi. Ide utama dibelakang CORBA adalah sebuah perangkat lunak perantara (intermedier) atau disebut middleware yang mengatur dan menyebarkan akses request ke sekumpulan data tertentu. Perangkat lunak perantara ini adalah middleware ORB (*Object Request Broker*). ORB melakukan interaksi dan membuat request ke berbagai macam objek. ORB ini terletak diantara layer data dan layer aplikasi dalam susunan arsitektur jaringan 7 layer model OSI. ORB akan melakukan negosiasi antara pesan *request* dari objek ke objek atau dari objek server ke sekumpulan data (data set). Tujuan CORBA adalah untuk membuat pemrograman lebih mudah dengan membuat aplikasi berbasis CORBA yang sangat *portable*.

Untuk melihat lebih detail tentang arsitektur CORBA dan ORB berada didalamnya, maka CORBA memiliki sebuah arsitektur yang disebut OMA (Object Management Group).OMA mengelompokkan jenis-jenis interaksi antar program untuk memudahkan penyediaan dukungan. Gambar 2.1 menggambarkan konsep OMA.



Gambar 2.1 Konsep Object Management Architecture (OMA)

OMA melakukan strukturisasi dunia aplikasi ke dalam dua kelompok besar: kategori layanan CORBA *(CORBAservices)* dan kategori fasilitas CORBA *(CORBAfacilities)*. Layanan CORBA menyediakan fungsi-fungsi dasar yang digunakan oleh hampir setiap obyek dalam berbagai aplikasi. Fungsi-fungsi ini biasanya bersifat generik dan tidak tergantung pada jenis domain aplikasi. Sebagai contoh adalah layanan penamaan *(naming service)*. Bayangkan bila memerlukan sebuah layanan tapi tidak tahu ke mana harus mencari *server* yang menyediakan layanan tersebut. Layanan penamaan dapat membantu layaknya sebuah "halaman kuning" *(yellow pages)*; dia bisa menyiarkan direktori layanan yang terdaftar padanya. Karena sifatnya yang generik, layanan penamaan dapat digunakan oleh aplikasi dari berbaga i domain.

Fasilitas CORBA menyediakan layanan pada level aplikasi. Ada dua jenis fasilitas: horizontal, yang diperlukan oleh berbagai jenis domain (misalnya, *user-interface*), dan vertikal, yang berlaku khusus untuk domain tertentu. Fasilitas horizontal fungsinya mirip dengan layanan CORBA, tetapi beroperasi pada level yang lebih tinggi karena berhubungan langsung dengan aspek fungsional dari aplikasi. OMG secara terus-menerus melakukan standarisasi terhadap *interface* untuk komponen-komponen di masing-masing kategori. Semakin banyak layanan dan fasilitas yang distandarisasi, semakin mudah untuk mencapai komputasi terdistribusi berbasis

komponen dalam berbagai bidang secara *plug-and-play*, tanpa terganggu oleh masalah heterogenitas.

2.3 Software CORBA-ORB

Ada banyak software CORBA-ORB baik yang free dan opensource maupun yang komersil. Yang akan digunakan disini adalah software ORB dari Java yang disebut Java-ORB dan JAC-ORB dari Fu Berlin / Xtradine. Keduanya berbasis Java. Sementara software ORB yang lainnnya dapat dilihat dalam tabel berikut:

Tabel 2.1 Tabel Software CORBA-ORB [4]

ORB Name	Product	Bhs Pemrogrmn	Licence
JAVA ORB	SUN MicroSystem	Java	Free
VisiBroker	Visigenic/Borland	Java, C++, Delphi	Evaluation
JacORB	Fu Berlin/XTRADyn	Java	Free
MICO	Mico.org/GPL Soft.	C++	Free
ISP	Rusian Univ	C++	Free
ORBacus	IONA	Java, C++	Free
OMNIORB	OMNI	Java, C++	Evaluation
VBORB		Visual Basic	Free
MTdORB	Phyton	Phyton	Free

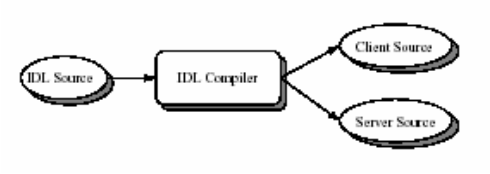
Ketika masuk ke pemrograman secara teknis maka yang perlu diketahui adalah sebuah bahasa interface yang digunakan untuk menghubungkan berbagai macam aplikasi. Bahasa ini disebut Interface Definition Language (IDL). Untuk mengenal tentang IDL maka berikut adalah pembahasan tentang IDL.

2.4 Interface Definition Language (IDL)

IDL merupakan inti untuk pembuatan aplikasi CORBA. IDL memuat sekumpulan tipe variabel yang kemudian akan dipetakan ke berbagai bahasa pemrograman.

Suatu *interface* dalam CORBA menyediakan sebuah deskripsi fungsi-fungsi yang disediakan untuk sebuah objek. Atribut, method dan parameter adalah informasi-informasi yang dispesifikasi oleh sebuah *interface*. IDL adalah sebuah bahasa yang menggambarkan objek-objek *interface* dalam sebuah aplikasi CORBA. [3]

IDL mendefinisikan tipe-tipe objeknya dengan mendefinisikan *interface-interface*nya. Sebuah *interface* terdiri dari sekumpulan nama operasi dan parameter-parameter untuk operasi-operasi tersebut. Catatan bahwa IDL hanya digunakan untuk menggambarkan *interface*, dan bukan sebuah implementasi. IDL bukan pula bahasa pemrograman. Melalui IDL, sebuah implementasi objek tertentu memberitahukan pada client tentang operasi-operasi apa saja yang tersedia dan memberitahukan bagaimana cara untuk mengambilnya. Dengan definisi IDL, objek-objek CORBA harus dipetakan ke beberapa bahasa pemrograman yang akan digunakan untuk membangun aplikasi client dan juga server. Beberapa bahasa pemrograman yang memiliki pemeta IDL (*IDL mapping*) adalah C, C++, Java, Smalltalk, Lisp, Basic, Pascal dan Python. Setelah mendefinisikan sebuah *interface* untuk objek-objek dalam IDL, *programmer* bebas untuk meng-implementasikan objek-objek dengan menggunakan bahasa pemrograman yang sesuai yang memiliki pemeta IDL. Sebagai sebuah protokol kompiler, IDL bisa digambarkan sebagai berikut:



Gambar 2.2 IDL *Compiler*

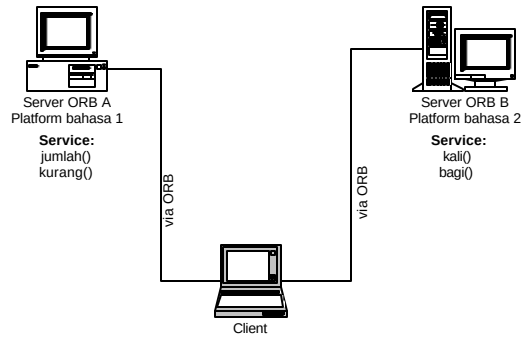
III. Implementasi Pemrograman Jaringan Menggunakan Middleware CORBA-ORB

Untuk membangun aplikasinya ada beberapa langkah yang perlu diperhatikan dalam hal pemrograman menggunakan middleware CORBA-ORB. Langkah-langkah itu adalah :

- Membuat rancangan dan analisa kebutuhan sistem
- Men-download software yang akan digunakan (software yang akan digunakan adalah Java-ORB dan JAC-ORB)
- Coding atau melakukan langkah-langkah teknis pemrograman

3.1 Analisa Kebutuhan

Untuk membangun pemrograman jaringan dengan Middleware CORBA-ORB, maka perlu dianalisa terlebih dahulu hal-hal yang diperlukan sebelum masuk ke langkah coding/pemrograman. Sesuai dengan tujuan yang ingin dibuat adalah aplikasi yang melakukan koneksi melalui ORB, dan bisa menunjukkan aplikasi yang terpisah dan berbeda platform, maka bisa digambarkan sebagai berikut :



Gambar 3.1 Rancangan Arsitektur Sistem Aplikasi

Dari gambar diatas terlihat bahwa ada 2 aplikasi yang jalan di dua platform pemrograman yang berbeda. Platform yang akan digunakan adalah Java-ORB dan Jac-ORB. 2 aplikasi tersebut juga akan diinstal pada platform sistem operasi yang berbeda. Sistem operasi yang akan digunakan adalah Linux untuk Jac-ORB dan Windows untuk Java-ORB. Dua aplikasi itu pun berada pada mesin yang berbeda. Disisi lain client pun akan menggunakan platform yang berbeda dengan server-nya. Jika ingin mengakses Server Java-ORB, maka yang akan digunakan adalah

aplikasi Client yang dibuat dengan JAC-ORB. Sedangkan jika client ingin mengakses aplikasi yang ada di platform JAC-ORB, maka akan digunakan aplikasi client yang dibangun dengan Java-ORB.

3.2 Coding : Implementasi Pemrograman

Beberapa langkah yang harus dilakukan untuk membangun aplikasi sesuai dengan arsitektur diatas adalah :

1. Membuat bahasa definisi IDL
2. Membuat aplikasi Server
3. Membuat Aplikasi Client
4. Mengkompilasi semua class yang telah dibuat

Contoh kasus ini adalah program jumlah() dan kurang() untuk server Java-ORB dan program kali() dan bagi() untuk server JAC-ORB. Maka, untuk bahasa IDL untuk aplikasi itu adalah sebagai berikut:

```
//idl Source Java-ORB
// file : javaidl.idl
module javaidl {
interface Iface_java {
float tambah(in float a, in float b);
float kurang(in float a, in float b);
};
};
```

```
//idl source JAC
//file : jacidl.idl
module jacidl {
interface Iface_jac {
float kali(in float a, in float b);
float bagi(in float a, in float b);
};
};
```

Dua file tersebut kemudian dikompilasi dengan menggunakan IDL Compiler yang dimiliki oleh Java-ORB dan JAC-ORB.
Langkah berikutnya membuat aplikasi server untuk kedua ORB yang digunakan.

```
//Java server source
//file : serverJava.java
import org.omg.CORBA.*;
import java.io.*;
import javaidl.*;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;

class servantPOA extends Iface_java {
public float tambah ( float a, float b) {
float c = a + b;
return c ;
}

public float kurang (float a, float b) {
float c = a- b;
return c;
}
}

public class serverJava {
public static void main (String[] args) {
try {

        ORB orb = ORB.init(args,null);
        servantPOA implRef = new
servantPOA();
        org.omg.CORBA.Object obj=
orb.resolve_initial_references("RootPOA");
        POA rootPOA = POAHelper.narrow(obj);
        rootPOA.the_POAManager().activate();
        org.omg.CORBA.Object objRef = null;
        objRef =
rootPOA.servant_to_reference(implRef);

        String ior =
orb.object_to_string(objRef);
        System.out.println("IOR : " + ior);
        FileWriter fw = new
FileWriter("java.ref");
        fw.write(ior);
        fw.close();
        System.out.println("Java Server:
operasi tambah()  ..... kurang()...");
        System.out.println("Server started");
        java.lang.Object sync = new
java.lang.Object();
        synchronized (sync) {
                sync.wait();
                }
        }
catch (Exception e) {
        System.err.println("error : " + e);
        e.printStackTrace(System.out);
        }
}
}
```

```
//*****JAC server source
//file : serverJAC.java

import org.omg.CORBA.*;
import java.io.*;
import jacidl.*;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;

class servantJac extends Iface_jac {
public float kali (float a, float b) {
float c = a * b;
return c;
}

public float bagi(float a, float b) {
float c = a / b;
return c;
}
}

public class serverJAC {
public static void main (String[] args) {
try {
        java.util.Properties props = new
java.util.Properties();
        props.setProperty("jacorb.implname","
StandardNS");
        servantJAC implRef = new
servantJAC();
        org.omg.CORBA.Object obj=
orb.resolve_initial_references("RootPOA");
        POA rootPOA = POAHelper.narrow(obj);
        rootPOA.the_POAManager().activate();
        org.omg.CORBA.Object objRef = null;
        objRef =
rootPOA.servant_to_reference(implRef);

        String ior =
orb.object_to_string(objRef);
        System.out.println("IOR : " + ior);
        FileWriter fw = new
FileWriter("jac.ref");
        fw.write(ior);
        fw.close();
        System.out.println("Server JacORB :
operasi kali()...operasi bagi()....");
        System.out.println("Server started");
        java.lang.Object sync = new
java.lang.Object();
```

```
        orb.run();
        }
catch (Exception e) {
        System.err.println("error : " + e);
        e.printStackTrace(System.out);
        }
}
}
```

Dua file server diatas sebelum diinstallkan di dua komputer server yang berbeda terlebih dulu di kompilasi.
Langkah berikutnya adalah membuat aplikasi client :

```
//*****client source Java & JAC ORB
//file : clientAll.java
import javaidl.*;
import jacidl.*;
import org.omg.CORBA.*;
import java.io.*;
import java.text.*;
import java.util.*;

public class clientAll {
public static void main(String[] args) {
try {
        BufferedReader baca = new
BufferedReader(new
InputStreamReader(System.in));
        float nil_a, nil_b;
        String dibaca_a, dibaca_b;
        System.out.println("\nMasukkan Nilai
yang akan di proses oleh tiap Server");
        System.out.print("Nilai A : ");
        dibaca_a =baca.readLine();
        System.out.print("Nilai B : ");
        dibaca_b = baca.readLine();

//Client JAVA
        FileReader fr1 = new
FileReader("java.ref");
        BufferedReader br1 = new
BufferedReader(fr1);
        String ior1 = br1.readLine();
        org.omg.CORBA.Object objRef1 =
orb.string_to_object(ior1);
        Ifacejava servantJava
=IfacejavaHelper.narrow(objRef1);

        System.out.println("\nJava ORB
Service... \n");
        nil_a =
Float.valueOf(dibaca_a).floatValue();
        nil_b =
Float.valueOf(dibaca_b).floatValue();
        System.out.println("Hasil Tambah() :
" + servantJava.tambah(nil_a,nil_b));
        System.out.println("Hasil Kurang() :
" + servantJava.kurang(nil_a,nil_b));
        System.out.println("\n");

//client Jac ORB
        FileReader fr2 = new
FileReader("jac.ref");
        BufferedReader br2 = new
BufferedReader(fr2);
        String ior2 = br2.readLine();
        org.omg.CORBA.Object objRef2 =
orb.string_to_object(ior2);
        Ifacejac servantJac
=IfacejacHelper.narrow(objRef2);
        System.out.println("JAC ORB
Service... \n");
        System.out.println("Hasil Kali() : "
+ servantJac.kali(nil_a,nil_b));
        System.out.println("Hasil Bagi() : "
+ servantJac.bagi(nil_a,nil_b));
        }
catch (Exception e) {
        System.out.println("error : " + e);
        e.printStackTrace(System.out);
        }
}
}
```

IV. Tinjauan Kritis tentang Pemrograman Memanfaatkan Middleware

Hal yang sangat menarik dari pemrograman dengan memanfaatkan middleware CORBA-ORB adalah isu tentang interoperasi, atau diistilahkan interoperabilitas. Isu ini memandang 2 sistem berbeda yang ingin melakukan komunikasi. Middleware CORBA-ORB bisa dijadikan suatu fasilitator untuk melakukan interoperasi itu. Middle ware CORBA-ORB juga bisa digunakan untuk penghubung dari banyak aplikasi yang berbeda platform bahasa pemrograman.
Isu yang lain yang juga terlibat dalam pemrograman middleware CORBA-ORB adalah masalah scalability. Isu ini dimaksudkan untuk penambahan / perluasan skala pengembangan aplikasi. Untuk membuat aplikasi jaringan client server tidak dibatasi dengan satu platform pemrograman saja tetapi bisa lebih luas lagi dengan berbagai platform pemrograman.
Isu yang juga menarik adalah kemudahan pengembangan. Dengan sistem terpisah-pisah secara fisik maupun logika, maka tim developer bisa dipecah-pecah secara mudah berdasarkan pemisahan itu. Ahli pemrograman Java bisa concern dengan aplikasi Java, ahli pemrograman C++ bisa concern dengan aplikasi C++, begitu juga programer Delphi dan lain-lainnya.

V. Kesimpulan

Beberapa hal bisa dijadikan benang merah dalam tulisan ini.

1. Middleware CORBA-ORB bisa membangun dua buah sistem untuk bisa saling melakukan interoperasi
2. middleware CORBA-ORB sangat membantu untuk pengembangan skala program aplikasi
3. Middleware CORBA-ORB dapat memudahkan untuk pengembangan tim developer.

Pustaka

[1] ---, www.webopedia.com/TERM/M/middleware .html
[2] ---, www.omg.org
[3] Mahmoud H., Qusay , *Distributed Programming With Java*, Manning, 2000.
[4] Somantri, Maman, *Pemrograman Lintas Bahasa Pemrograman dalam Arsitektur CORBA*, Prosiding Seminar Nasional Teknologi Informasi, STTNAS Yogyakarta, Juni 2005.