The Distributed Programming Environment on the Internet

Chang-Hyun Jo*, Jea Gi Son, Younwoo Kang, and Phill Soo Lim
***Department of Computer Science, University of North Dakota**
Department of Computer Science, Kyonggi University, Korea
jo@cs.und.edu

KEYWORDS
Distributed Programming Environment, Internet, Distributed Objects, CORBA, Java, CSCW

ABSTRACT

A computing using distributed objects provides a flexible and scalable programming on the distributed and parallel programming environment. People work together on a scientific research, scholarly work, and software development with help from computers in the environment of the computer-supported cooperative work (CSCW). The aim of this research work is to develop the cooperative programming environment - namely the Distributed Programming Environment (DPE) - supporting the distributed object computing on the distributed system such as the Internet. The system has been developed based on the CORBA/Java on the Internet. This paper explains precisely our experience of the implementation and experimentation.

1. INTRODUCTION

Recently due to the advances of the computing facilities and the development of the network in which the computing resources are connected and shared, the programming environment migrates from the single and local environment to the distributed and global environment. Therefore the programming environment that can smoothly support cooperative works on the network is necessary.

A computing using distributed objects provides a flexible and scalable programming on the distributed and parallel programming environment. Many systems such as CORBA, DCOM, and Java support the distributed object computing.

--
ACM SAC 1999, San Antonio, Texas, 85-90, (Feb.28-March 2, 1999).

People work together on a scientific research, scholarly work, and software development with help from computers in the environment of the computer-supported cooperative work (CSCW) [Palmer and Fields 1994]. CSCW is also known as computer-supported cooperation, GroupWare, Workflow, and Group Decision-Support Systems. To bring CSCW into the real use, Palmer (1994) has pointed out that the following considerations should be solved:

- Developing user-friendly software,
- Addressing the social dynamics of group activities,
- Standardizing various terms, and
- Handling the interactions between multiple tasks.

Among these research works to be done, the most important issue for adopting CSCW successfully is to develop an environment in which the users can collaborate with their work easily and friendly.

The aim of this research work is to develop a cooperative programming environment – namely Distributed Programming Environment (DPE) – in which programmers geographically distributed all over the world can develop and manage software conveniently while supporting the distributed object computing on the distributed system such as the internet. This research work is to develop a programming environment that supports an efficient programming on the distributed environment. The result has been expected to support the following criteria:

- The cooperative programming environment on the distributed systems
- The environment to support the program development using the Internet

The integrated development environment on the distributed system may allow the smooth communication between programmers located remotely, and may reduce the total software development cost and time by decreasing the costly meetings. A framework for the distributed objects provides a higher level of both interoperability and transparency among the objects resided in the distributed nodes.

Java supports the mobile codes needed for the mobile agents in a distributed system [Java 1998]. Mobile codes can be executed dynamically by distributed applications. Java packages allow dynamic class linking with methods that can be overridden at run time. The language features in Java help programmers to write client/server components that can interoperate easily. Java also provides a high level of security and reliability in developing the distributed environment.

OMG proposes the Common Object Request Broker Architecture (CORBA) as a standard for supporting distributed objects [Orfali and Harkey 1998]. While the traditional objects reside in a single processor (within a single process or multiple processes), distributed objects may reside in several nodes in a network. Robust distributed objects may be written in different languages, and can be compiled by different compilers, and communicates each other via standardized protocols embodied by middleware [Lewandowski 1998]. CORBA-based computing has been a typical research topic for the past few years.

The collaborative works and tools are very useful for software engineering and general science research works [Kouzes et al. 1996] [Palmer and Fields 1994]. Several research works in distributed computing environment have been reported.

Wulf introduced the term - “collaboratory” that is merged two words - collaboration and laboratory [Kouzes et al. 1996]. A collaboratory is a cooperative laboratory that provides a research environment interacting with colleagues regardless geographical location, accessing instrument, sharing data and computational resources. There have been many prototypes of such collaboratories [Kouzes et al. 1996].

The DPE is a CSCW system that combines the distributed computing environment and the collaborative software development.

The section 2 presents overall design of the DPE. Section 3 reports our implementation scheme. A programming example with DPE is shown in section 4, and we conclude in section 5 with the references.

2. DISTRIBUTED PROGRAMMING ENVIRONMENT

2.1 Distributed Programming Environment

The Distributed Programming Environment (DPE) is designed and implemented to support an environment for distributed programming on the parallel and distributed programming systems. The DPE implementation is currently working on the Internet. The DPE allows several programmers, project managers, and system managers to cooperate each other, while they are residing on the different sites all over the world. Cooperative members can work anywhere, even while they are on business travel carrying with notebook computers, by using the DPE with any machine if the machines are connected by the Internet and supported by CORBA/Java.

The DPE supports the four modes in CSCW [Palmer and Fields 1994]:

(1) Synchronous – People may be in the same location at the same time working together,
(2) Distributed synchronous – Cooperative activities take place at the same time but participants are located at the different sites,
(3) Asynchronous – Activities take place at different times but in the same location, and
(4) Distributed asynchronous - Activities take place at different sites at different time.

The DPE supports also the following computing schemes:

(1) Distributed Programming – Programmers can be geographically and logically distributed anywhere, that means, the organization of software firm may be more flexible and dynamically configurable.
(2) Distributed Managing – Managers can be located anywhere freely while they can still manage formally by using portable computers. The DPE system management is also possible remotely.
(3) Mobile Computing – Computing facilities can be more dynamically configurable. Load balancing and mobile computing is possible.
(4) Security – More robust security system can be also supported by adding an appropriate security scheme.

Several kinds of collaborations in the DPE are possible like the following:

(1) Server-Client
(2) Manager-Programmer
(3) Programmer-Programmer
(4) Supplier-User

The advantages of the DPE is like the following:

(1) Easy Access to Distributed Computing Environment
(2) Smart User Interfaces
(3) Machine Independent
(4) Project Management
(5) Remote Management
(6) Remote Access to Distributed Resources

(7) Mobility
(8) Load Balancing
(9) Stability and Flexibility
(10) Information Management
(11) Internet Communication Technologies
(12) Integration of Heterogeneous Systems

2.2 Object-Oriented Modeling for the DPE

The object-oriented modeling for the DPE is partially developed using the standard object-oriented modeling language – UML based on the incremental and iterative development process.

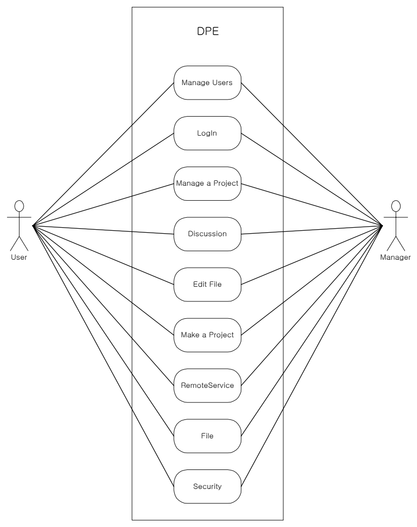
A use case is a typical interaction between a user and a computer system [UML 1997] [Larman 1998]. A use case captures some user-visible function, and achieves a discrete goal for the user. The use case is an essential tool in requirement capture and in planning and controlling an iterative project [Fowler 1997]. An actor is a role of object or objects outside of a system that interacts directly with it as part of a coherent work unit (a use case). An actor element characterizes the role played by an outside object; one physical object may play several roles and therefore be modeled by several actors. The following figure shows a use case – "Manage a Project". The use case includes an explanation of an actor – "Manager" who manages some projects in the system [Figure 1].

Use Case: Manage a Project
Actors: Manager
Purpose: Manage a project.
Type: Primary
Description: A manager request a project management upon the system. The system validate the privilege of the manager by asking a proper password to the project. If the manager is valid, then the appropriate project will be opened, otherwise the request is rejected.
Cross References: Functions: R1.1, …
Use Cases: Manager must have completed the "Log In" use case.

<Figure 1> Use Case

A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) associations between the actors and the use cases, and generalizations among the use cases [UML 1997]. A use case describes interaction with a system. The DPE system has been chosen for the system boundary [Figure 2].

A class represents a concept within the system being modeled. From the use cases several classes can be extracted to generate appropriate components for the DPE system.

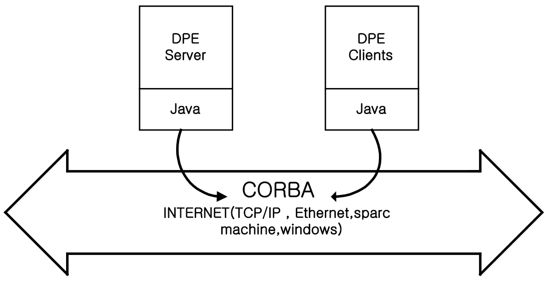


<Figure 2> Use Case Diagram

3. THE IMPLEMENTATION OF DPE

3.1 Implementation Environment

The DPE (version 1.0) is implemented by using CORBA/Java on the Internet [Figure 3]. The DPE can be working on any system with CORBA/Java. The DPE consists of a server component and a client component. A client component can be distributed several nodes. Many programmers and managers can use their own client's copies on their machine.



<Figure 3> An Implementation Environment

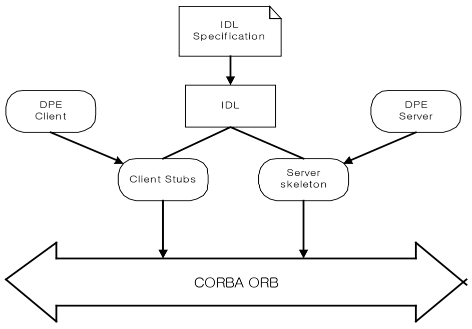
3.2 Java

Java has a mobile code system [Java 1998] [Deitel 1998]. Java simplifies code distribution in a network supporting CORBA. Java deals with code transparency, while CORBA supports network transparency. The current implementation has been compiled by using the JDK 1.1.6 [Java 1998]. Java's AWT provides an excellent scheme of graphical user interfaces. It reduces a lot of time to dedicate to the GUI-related code. The current implementation uses the Java's Swing packages to produce a nice user interface. The Just-in-time (JIT) compilers may help Java to increase its execution speed.

3.3 CORBA

Common Object Request Broker Architecture (CORBA) is the most famous middleware supported by a consortium of more than 800 companies [Orfali and Harkey 1998]. The consortium represents the most of the computing companies except Microsoft, which has its own object broker called Distributed Component Object Model (DCOM). CORBA object bus defines the object components and their interoperability. The Object Request Broker (ORB) is the object bus. CORBA is designed to allow object components to discover each other and to interoperate on an object bus. CORBA supports a transparent object references among distributed objects through object interfaces.

A CORBA ORB integrates Java objects smoothly in the distributed computing environment, while providing many common object services [Figure 4]. CORBA objects are packaged as binary components, and resided anywhere on a network. The remote clients can access server objects via method invocation, without knowing in where the distributed objects reside or on which operating system it executes on. We have used the CORBA 2.0 specification [Orfali and Harkey 1998] [Vogel and Duddy 1997], and Borland/Visigenic's VisiBroker for Java 3.2 [VisiBroker 1998].



<Figure 4> CORBA Architecture

One of the benefits from CORBA ORB is to provide static and dynamic method invocations. An ORB allows to define statically a method invocation at compile time, or to discover dynamically at run time. It provides either strong type checking or flexible dynamic binding.

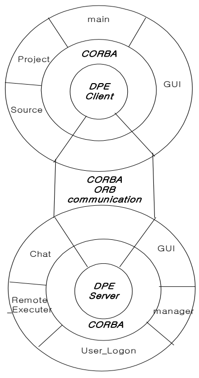
CORBA needs to specify the interfaces. The CORBA specification is written in Interface Definition Language (IDL). The IDL does not show any implementation details. The IDL grammar is a subset of C++ with additional keywords to support distributed objects, while supporting standard C++ preprocessing. Here is an example IDL for the client and server parts of the DPE system [Figure 5]. In this example, the module for connection – 'DPEConnection' defines the interfaces for client and server - 'DPEClient' and 'DPEServer'. Both interfaces include several services for login, source access, message, remote compilation and execution, remote file access, and chatting.

```
module DPEConnection {
    interface DPEClient {
            string login( in string userName, in string password );
            string registerSourceClient( in string fileName );
            string removeSourceClient( in string fileName );
            string clientInsertMessage( in string fileName, in long offs, in string str );
            string clientRemoveMessage( in string fileName, in long offs, in long len );
            void serverInsertMessage( in string fileName, in long offs, in string str );
            void serverRemoveMessage( in string fileName, in long offs, in long len );

            string getRemoteFile( in string fileName );
            string putRemoteFile( in string fileName, in string text );
            string remoteCompile( in string command, in string fileName );
            string remoteRun( in string command, in string fileName );
            string registerChatClient( in string fileName );
            string removeChatClient( in string fileName );
            string sendChatMessage( in string msg );
            void serverChatMessage( in string msg );
     };

     interface DPEServer    {
            string login( in DPEClient clientObj, in string clientObjStr, in string userName, in string Password );
            string getFilePermission( in string clientObjStr );
            string registerSourceClient( in string clientObjStr, in string fileName );
            string removeSourceClient( in string clientObjStr, in string fileName );
            string insertSourceMessage( in string clientObjStr, in string fileName, in long offs, in string str );
            string removeSourceMessage( in string clientObjStr, in string fileName, in long offs, in long len );
            string getRemoteFile( in string clientObjStr, in string fileName );
            string putRemoteFile( in string clientObjStr, in string fileName, in string text );
            string remoteCompile( in string clientObjStr, in string command, in string fileName );
            string remoteRun( in string clientObjStr, in string command, in string fileName );
            string registerChatClient( in string clientObjStr, in string group );
            string removeChatClient( in string clientObjStr, in string group );
            string chatMessage( in string clientObjStr, in string group, in string msg );
     };

};
```

<Figure 5> An IDL for the DPE

3.4 How To Work Together

A combining CORBA and Java provides a notable solution to implement a collaborative software development environment on the distributed system. CORBA/Java provides much simpler way to generate client/server applications [Deitel 1998] [Orfali and Harkey 1998]. The interfaces in Java are similar to the IDL interfaces in CORBA. They both specify the services of an object without showing any implementation.



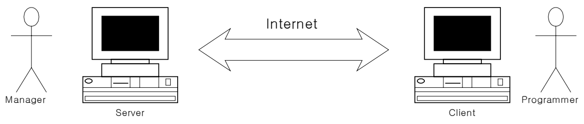
<Figure 6> Distributed Programming Environment using CORBA and Java

The Java implementation of client part includes client/main control, client-GUI, project management, and source management [Figure 6]. The Java implementation of the server part consists of server-GUI, management, user-login, remote execution, and chatting.

4. AN EXAMPLE

4.1 An Example Scenario

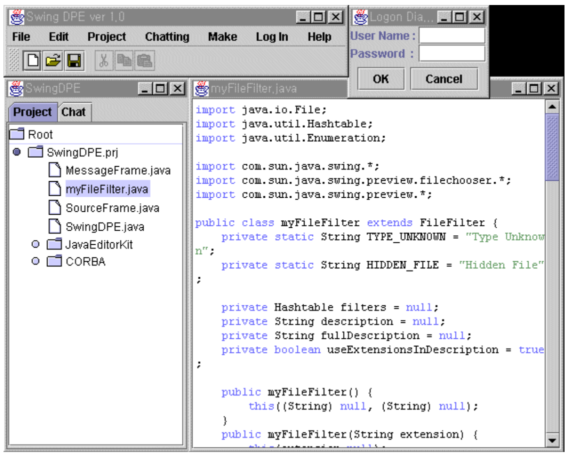
A scenario of our example session is as follows: A manager and a programmer have located in separate clients' nodes in a network [Figure 7]. The manager wants to open, compile, and execute a program that has been submitted by the programmer to get an approval from the manager. The programmer may look at a window that shows the process of compilation and execution run by manager. There is another window for chatting among them to discuss and comment their work during the session.



<Figure 7> An Example Session

4.2 Programming Environment

The programming environment in the DPE is shown in the following figure [Figure 8]. Users may meet the log-in prompt to be allowed to enter the DPE system. The system manager may control the user's information. Each user has an access right to the files and commands specified the uses' information file. Each client/server in the network has the same user interface to the DPE. The DPE manager may reside anywhere geographically and logically, and may control the system using its own authority.



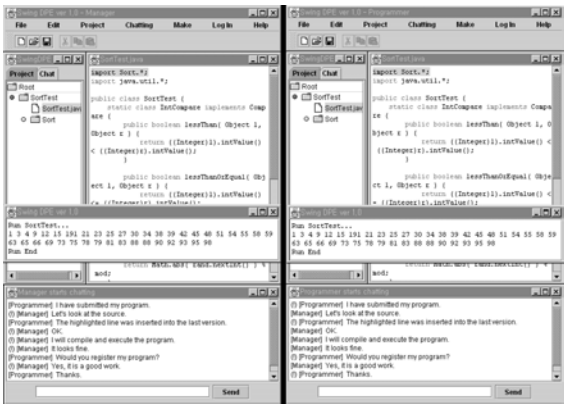
```
import java.io.File;
import java.util.Hashtable;
import java.util.Enumeration;

import com.sun.java.swing.*;
import com.sun.java.swing.preview.filechooser.*;
import com.sun.java.swing.preview.*;

public class myFileFilter extends FileFilter {
    private static String TYPE_UNKNOWN = "Type Unknow
n";
    private static String HIDDEN_FILE = "Hidden File"
;

    private Hashtable filters = null;
    private String description = null;
    private String fullDescription = null;
    private boolean useExtensionsInDescription = true
;

    public myFileFilter() {
        this((String) null, (String) null);
    }
    public myFileFilter(String extension) {
```

<Figure 8> A Programming Environment

4.3 How It Works

An example session is shown in the following figure [Figure 9]. A manager and a programmer are located in the clients' nodes separately on the Internet. Some overlapped windows for the manager are shown at the left-hand side, and the other windows at the right-hand side are for the programmer. The windows for the DPE menu, project management, source code, execution and chatting are overlapped at each side.

Then the manager and programmer are ready to see, compile, and execute the source program, while chatting each other. With the DPE, they may discuss about the program, while looking at the source and the result of compilation and execution of the program. As the programmer reports his submission of a program to get permission for the program registration, the manager wants to see the source and test it, while the programmer looks simultaneously at the processing.


Swing DPE ver 1.0 - Manager
File Edit Project Chatting Make Log In Help
SwingDPE
Project Chat
Root
SortTest
SortTest.java
Sort
SortTest.java
Swing DPE ver 1.0
Manager starts chatting
Send
Swing DPE ver 1.0 - Programmer
Programmer starts chatting

```
import Sort.*;
import java.util.*;

public class SortTest {
    static class IntCompare implements Comp
are {
        public boolean lessThan( Object l,
Object r ) {
            return ((Integer)l).intValue()
< ((Integer)r).intValue();
        }

        public boolean lessThanOrEqual( Obj
ect l, Object r ) {
            return ((Integer)l).intValue()
<= ((Integer)r).intValue();
```

```
Run SortTest...
1 3 4 9 12 15 191 21 23 25 27 30 34 38 39 42 45 48 51 54 55 58 59
63 65 66 69 73 75 78 79 81 83 88 88 90 92 93 95 98
Run End
```

```
        return Math.abs( rand.nextInt() ) %
mod;
```

[Programmer] I have submitted my program.
(!) [Manager] Let's look at the source.
[Programmer] The highlighted line was inserted into the last version.
(!) [Manager] OK.
(!) [Manager] I will compile and execute the program.
(!) [Manager] It looks fine.
[Programmer] Would you register my program?
(!) [Manager] Yes, it is a good work.
[Programmer] Thanks.

(!) [Programmer] I have submitted my program.
[Manager] Let's look at the source.
(!) [Programmer] The highlighted line was inserted into the last version.
[Manager] OK.
[Manager] I will compile and execute the program.
[Manager] It looks fine.
(!) [Programmer] Would you register my program?
[Manager] Yes, it is a good work.
(!) [Programmer] Thanks.

<Figure 9> A DPE Example

As we can see in the example session, the manager and programmer may reside at the logically and geographically different locations, and they can cooperate with each other. The manager may also manage remotely his work wherever he is if he can access the DPE on the Internet. The system manager of the DPE can even take care of the system wherever he is. The cooperative work for the software development can be more efficient with the DPE.

5. CONCLUSION

The DPE system provides an environment for distributed programming on the Internet. It may reduce the total cost of software development by reducing the cost of time and space of programmers distributed on the network. The DPE cannot be used only for the integrated programming environment on the distributed system, but can be applied also to the applications such as teleconferencing and cyber lectures.

The extensions to the DPE to robustly support security, heterogeneous programming languages, and multimedia contents, and more graphical cooperation are the future works.

REFERENCES

[1] Visibroker for Java, VISIGENIC Reference Manual, V.3.2, "http://www.visigenic.com", (1998).

[2] Deitel, H. M., and P. J. Deitel, Java: How to Program, Prentice Hall, (1998).

[3] Fowler, M. UML Distilled: Applying the Standard Object Modeling Language, Addison Wesley, (1997).

[4] Java Developer's Kit (JDK), "http://www.javasoft.com", (1998).

[5] Kouzes, R. T., J. D. Myers, and W. A. Wulf, Collaboratories: Doing Science on the Internet, IEEE Computer, 29(8), 40-46, (August 1996).

[6] Larman, C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design, Prentice Hall, (1998).

[7] Lewandowski, S. M. Frameworks for Component-Based Client/Server Computing, ACM Computing Surveys, 30(1), 3-27, (March 1998).

[8] Orfali, R. and D. Harkey, Client/Server Programming with Java and CORBA, (2nd Ed.), John Wiley & Sons, Inc., (1998).

[9] Palmer, J. D. and N. A. Fields, Computer-Supported Cooperative Work: Guest Editors' Introduction, IEEE Computer, 15-16, (May 1994).

[10] UML Notation Guide, V.1.1, "http://www.rational.com/uml", (Sept. 1, 1997).

[11] Vogel, A. and K. Duddy, Java Programming with CORBA, John Wiley & Sons, Inc., (1997).