웹상에 분산된 시뮬레이션 객체들의 통합을 위한 시뮬레이션 모델링 방법론*

Simulation Modeling Approach for Integrating Distributed Simulation Objects on the Web

이영해**, 심원보**, 김숙한***, 김서진****
Young Hae Lee, Wonbo Shim, Sook han Kim, Seojin Kim

Abstract

The cost of simulation modeling, the expertise required, and the pains of starting a new each time are impediments to more wide spread adoption of simulation technology. In addition, one of the most critical problems in the field of computer simulation today is the lack of published models and physical objects within the World Wide Web (WWW) allowing such distribution. From the viewpoint of WWW as distributed model repositories, it can be assumed that very many simulation models exist on the web. This paper is based on the premise that WWW is a distributed repository.

Design Pattern, web-oriented technology like Java and CORBA, which are especially to cope with distributed objects, are introduced and discussed in detail for integration of simulation model. In this paper an architecture of model integration is proposed, which presents the whole procedure of model integration and how the Internet technologies are connected in. The central focus of this research is on the technical realization of integrating simulation models as distributed objects

* This work is supported in part by the Ministry of Information & Communication of Korea ("인터넷에서 운용 가능한 시뮬레이션 기술에 관한 연구<'98> " supervised by IITA
** 한양대학교 산업공학과
*** 한국오라클 consultant
**** 쌍용정보통신

1. Introduction

Recently the World Wide Web definitely represents a fertile area in which to perform computer simulation research. So web-based simulation became one of the most interesting fields of simulation researches today. Web -based simulation can be defined as the integration of internet technology with the field of computer simulation to enable execution on the web and to perform more efficient and effective simulations as new simulation methodology. There are several directions that one can take in creating a marriage between the web and simulation field. Fishwick offers a perspective on the issue of web-based simulation[5]. He identifies many potential impacts of web-based simulation technologies, with emphasis on education and training, publication and simulation programs. Since then, Page presented a literature review on web-based simulation and suggested five areas of focus:

i. simulation as hypermedia,
ii. simulation research methodologies,
iii. web-based access to simulation programs,
iv. distributed modeling and simulation,
v. simulation of the WWW[14].

Especially in research area of distributed modeling and simulation, two directions of research involve:

1) parallel and distributed model execution (or parallel and distributed simulation; PADS) on the web, and
2) distributed model repositories[6].

This paper focuses more on the area of distributed model repositories since there has been less research in the area than in the more mature field of PADS on the web. The concept of model repository lends itself to the study of how to organized model information. Since the web is also concerned with how to effectively organize information, this appears to be reasonable and natural way to blend the web with simulation.

Creating comprehensive simulation models can be expensive and time consuming. The web represents the future of information sharing and exchanging, and it is going to be envisioned an 'object metaphor' where even document is one type of object transferred 'document/desktop metaphor'. Nevertheless one of the most critical problems in the field of computer simulation today is the lack of published models and physical objects within a medium - such as the World Wide Web - allowing such distribution. From the viewpoint of Web as distributed model repositories, it can be assumed that very many simulation models exist on the web. Actually discussions and promotions of the publication and standardization of simulation objects stored on the web to enable model and object reuse are being emerged(Fishwick 1998, Page *et al.* 1999). Therefore on the premise that most simulation models are already stored at certain sites on the web, this paper discusses how to integrate the distributed simulation sub-models as objects for constructing the required simulation model which may be larger and more complex.

There are some issues related with using these distributed simulation objects on the web as a infrastructure of repositories:

1) how to find appropriate models on the web,
2) how to integrate the distributed objects which may be mostly heterogeneous,
3) formalization information of models or

objects which is dispersed over the web, and

4) how to support the execution of web-based simulation.

By Fishwick, the concept of Digital Object has been introduced. He insisted that there is needed to be a infrastructure or agreed-upon standards for true digital object engineering. For this, there are some implication to toward standards for the digital objects.:

1) the Unified Modeling Language (UML),
2) the High Level Architecture (HLA),
3) the VLSI Hardware Definition Language (VHDL) [7][8].

In this research, several web-oriented approaches are adopted for overcoming above problems. The first approach is based on the principles of object-oriented design concept, especially Design Patterns. Using design patterns as a shared language for communica -ting insight and experience about their problems and solutions, one can take great advantages that design patterns allow programmers to collaborate and combine their wisdom more effectively and also enable to extend the reusability of the previous made objects and their design. The second one is to use Java and CORBA technology. Today Java is the most powerful and popular language implementing web-based simulation. It provides two key capabilities which are portability and automatic software distribution. These portable and distributable capabilities are corresponding with mechanism of object distribution and integration. Java Interface Definition Language (IDL) gives very useful technology to solve the above mentioned problems because of using the many advantages of CORBA for dealing with distributed objects. Besides, a few concepts are introduced; the concepts of plug and play, adapters like capsules for putting together heterogeneous distributed objects, and data mining algorithm for searching the most suitable simulation model, a three -tiered client/server architecture which is constituted of user interface (presentation component), application logic (business rules component), and data management.

This paper concerns how one can treat with probably heterogeneous and non -standardized objects which are distributed on the web and how one can integrate them together for building a simulation model. Section 2 introduces the elements for dealing with distributed objects on the web. CORBA, Java IDL, and design patterns are mentioned. In section 3 the environment of integrating distributed simulation models as objects is described. Basically three tired structure are provided and the concept of agent is introduced. Section 4 describes the proposed architecture that provides the way of integrating of distributed simulation objects. Finally a prototypical model which is simply two node network model is presented to demonstrate the possibility and the capability of this approach.

2. FOUNDATIONS OF DEALING WITH DISTRIBUTED OBJECTS ON THE WEB

2.1 CORBA [2][17]

CORBA (Common Object Request Broker Architecture) is the standard distributed object architecture developed by the Object Management Group (OMG) consortium. This standard allows CORBA objects to invoke one

another without knowing where the objects they access reside or in what language the requested objects are implemented. The OMG -specified Interface Definition Language (IDL) is used to define the interfaces to CORBA objects.

CORBA objects differ from typical programming language objects in these ways:

1) CORBA objects can be located anywhere on a network,
2) CORBA objects can inter-operate with objects on other platforms,
3) CORBA objects can be written in any programming language for which there is a mapping from OMG IDL to that language (Mappings currently specified include Java, C++, C, Smalltalk, COBOL, and Ada).

The figure 1 shows a method request sent from a client to a CORBA object implementation in a server.

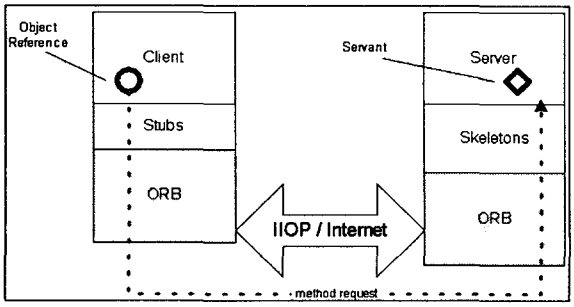

Figure 1. Implementation between client and server

The client has no knowledge of the CORBA object's location, implementation details, nor which ORB is used to access the object. Different ORBs communicate via the OMG-specified Internet InterORB Protocol (IIOP).

2.2 Java IDL (JIDL) [17]

Today Java is the most powerful and popular language for implementing web-based simulation. Moreover Java IDL gives a great chance to incorporate with CORBA.

Java IDL is an ORB provided with the JDK 1.2. It is a technology for distributed objects - that is, objects interacting on different platforms across a network. JIDL is similar to Remote Method Invocation (RMI). However, JIDL enables objects to interact regardless of whether they're written in Java programming language or another language such as C++. This is possible because Java IDL is based on the CORBA. JIDL supports the mapping for Java as each language that supports CORBA has its own IDL mapping. Together with the idltojava compiler, it can be used to define, implement, and access CORBA objects from the Java programming language. Java IDL is compliant with the CORBA/IIOP 2.0 Specification and the IDL-to-Java Language Mapping.

The Java IDL ORB supports transient CORBA objects - objects whose lifetimes are limited by their server process's lifetime. Java IDL also provides a transient name server to organize objects into a tree -directory structure.

The brief steps through the process of designing and developing a distributed object application with JIDL is the following: 1) Define the remote interface, 2) Compile the remote interface, 3) Implement the server, and then 4) Implement the client.

An IDL interface declares a set of client accessible operations, exceptions, and typed attributes (values). Each operation has a signature that defines its name, parameters,

result, and exceptions. For example, a simple IDL interface that describe the common machine's operation, follows.

```
module Station
{
    interface Queue
    {
        int capacity();
    };
    interface Activity
    {
        int delay();
    };
};
```

2.3 Design patterns [10]

Our approach for overcoming these problems is based on the principles of object -oriented design concept, especially Design Patterns. A definition which more closely reflects its use within the patterns community is: A pattern is a named nugget of instructive information that captures the essential structure and insight of a successful family of proven solutions to a recurring problem that arises within a certain context and system of forces. Due to using design patterns as a shared language for communicating insight and experience about their problems and solutions, one can take great advantages that design patterns allow programmers to collaborate and combine their wisdom more effectively and also enable to extend the reusability of the previous made objects and their design.

Patterns are usually described using a format that includes the following information even though pattern books, so called 'catalog', in how patterns present this information:

1) A description of the problem that includes a concrete example and a solution specific to the concrete problem.
2) A summary of the considerations that lead to the formulation of a general solution.
3) A general solution.
4) The consequences, good and bad, of using the given solution to solve a problem.
5) A list of related patterns.

So many design patterns are specified in a catalog of reusable design patterns. Some category of patterns are fundamental design patterns, creation patterns, partitioning patterns, structural patterns, behavioral patterns, concurrency patterns, and so forth. On this research several patterns, which are the most related to build and integrate simulation models, are particularly introduced later.

For example (See the Figure 2), Source node, which is derived from Node class, can be established in the following pattern so called Strategy. For presentation of the design patterns, we use the Unified Modeling Language (UML)[4].

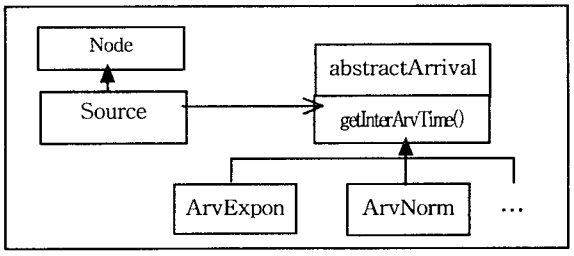

Figure 2. An example of Strategy pattern

3. AN ENVIRONEMT OF INTEGRATING DISTRIBUTED SIMULATION MODELS AS OBJECTS

3.1 Multi-tiered application[9]

Traditional applications are, for the most part, self-contained monolithic programs which have limited access to one another's procedures and data. They are usually cumbersome to build and expensive to maintain because even simple changes require the entire program to be recompiled and re-tested.

By contrast, CORBA-compatible application using distributed objects is made up of three -tiered architecture and forests a neat separation of concerns based on the Model /View/Controller (MVC) paradigm. It has a user interface code layer, a computation (simulation code or logic) layer, and a database access layer. All interactions between the layers occurs via the interfaces that all CORBA objects must publish. The figure 3 illustrates the three-tiered, modular applications.

3.1.1 user-interface(UI) tier

The UI tier usually invokes methods on the business logic tier and thus acts as a client of the simulation server on the web.

3.1.2 service (server, application logic) tier

This layer has a simulation objects - CORBA-compliant objects that perform logical simulation methods such as random number and variates generator, statistics object, report object.

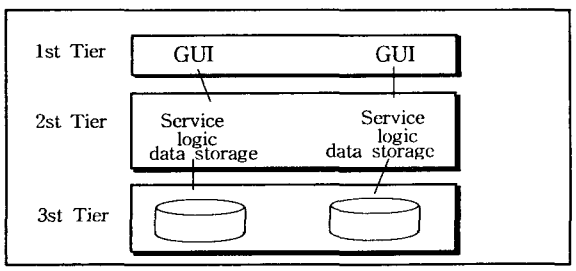

Figure 3. Three-tiered application

3.1.3 data store (Database) tier

The data store layer is made up of objects that encapsulate database routines and interact directly with the WEBDB. Some method in this application layer can deal with database SQL query.

3.2 Integrating distributed simulation objects

3.2.1 simulation models as object [15][16]

The simulation objects may have a broad meanings such as simple classes or their objects, Java Beans, components, even simulation sub-models. For instance, if you were creating a model of a manufacturing plant you might want to represent the machines, routings, work in process, the tools and fixtures, the customer orders and the workers. Each of these can be expressed by unit of object. So each object has its own state and functionality.

3.2.2 building a simulation model [3]

Creating comprehensive simulation models can be expensive and time consuming. It is worth while to develop a general methodology or environment that will allow simulation users to quickly and efficiently create high fidelity simulation models by linking independent model objects distributed across the Internet.

Basically we tried to apply design patterns to constitute simulation libraries. As the previous mentioned, design patterns give us great understanding and reusability of model objects. So it will allows us to way of integration many dispersed models. With all of the above, the integration of distributed simulation models is based on four fundamental themes: 1) models are objects, 2) they communicate with one another in client

/server relationships by mess-age-passing, 3) each model is represented by an agent that explains the capabilities of the model and assists with integration of that model, and 4) each distributed model can be implemented in parallel simulation.

3.2.3 agents [11]

The purpose of model agents is to facilitate the construction of network models. The roles of model agents are to be reactive with client as a simulation model authors, to find the appropriate model objects on the web, and to provide a interface to object for communication with other objects. For these aspects, model agents know what their model object can accomplish, what data they need to perform those actions, and what information the model will provide as it executes.

Figure 4 shows how the author to construct the simulation model by react with agents and simulation server.

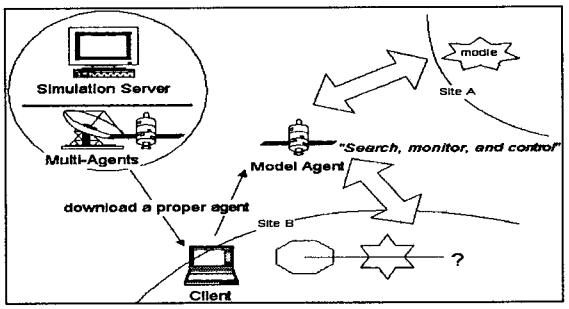

Figure 4. Interaction between simulation and model agent

3.3 Algorithm of integrating objects for the agent

An agent will help model builders select the appropriate model from among various models, and it will assist in configuration and operation of the distributed model networks. An important aspect of model integration is the selection of appropriate models to be linked.

However actually we do not deeply concern about the intelligence of model agent because it slightly step aside from the main focus of this paper.

3.4 Object-oriented Simulation [1][12][13]

In the object-oriented world view, objects are dynamically allocated data structures coupled with routines, called methods. The fields in the object's data structure define its state at any instant in time while its methods describe the actions which the object can perform. Other objects can query the value of an object's fields or ask it to perform its methods by sending messages to the object. Instead of invoking an object's methods by call, the user invokes the method by sending a message to the object requesting it to perform the method.

The need to write good simulation models was responsible for the development of another popular language with object oriented capabilities. The question is not how object oriented programming(OOP) can be used to help write better models, it is really how this simulation and modeling tool can be applied in other situations. So using OOP to support the writing of a model is simply an extension of the system being modeled. It makes the writing of the model easier and more natural.

Describing a system as collection of interacting components or objects is a natural way of breaking down any problem, large and small. Each object module describes the object behaviors, called methods. In addition, the characteristics of the object are contained

in its fields or variables. Everything the object knows is in its fields; and everything it can do is in its methods. This structure maps well to real world objects.

The re-use of libraries of pre-built objects holds out the promise of real productivity gains in software development. Without a means to adapt such objects to special needs, this promise is rarely fulfilled. The extensibility offered by inheritance, coupled with the modular separation of interface definitions from actual implementation code are the mechanisms needed to support practical re-use of object libraries. Object orientation offers other benefits to model development. The controlled access to object data structures through the object methods is just what is needed to build robust objects which can be the basis of re-use.

4. AN ARCHITECTURE FOR MODEL INTEGRATION AND PARALLEL DISTRIBUTED SIMULATION

The procedure of implementing the proposed web-based simulation modeling and performing parallel simulation is summarized in the followings steps.

Step 1.

A client who tries to make a simulation model connects the simulation server on the remote site.

Step 2.

The client can request a modeling task to a model agent. So one can construct a desired simulation model by interacting the agent.

Step 3.

The agent assists the client to find appropriate simulation sub-models or objects and monitors them.

Step 4.

Once building a required simulation model is complete, the client makes a request for performing the simulation.

Step 5.

Using the parallel distributed simulation on the web or Using the PDES system on a site is the other matter of decision making.

Alternatively the client simply download the required engines from the simulation server to run the simulation task by oneself.

Figure 5 shows how to implement integrating for constructing simulation model and to perform simulation in the web environment.

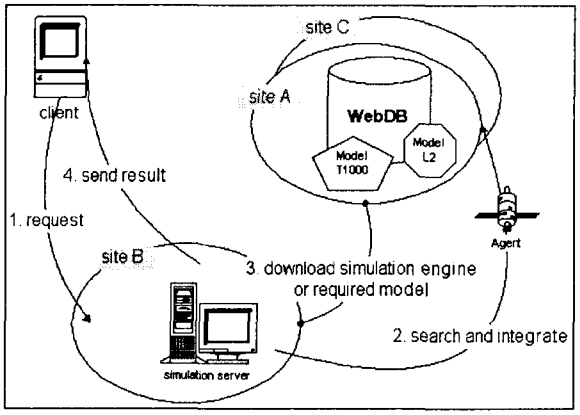

Figure 5. How to implement integrating for constructing simulation model and to perform simulation

The Internet itself, holds a massive parallel processing power in building simulation models and performing simulation itself.

4.1 The architecture of integration of distributed simulation objects

Figure 6 shows the whole architecture of integration of distributed simulation objects.

First of all this paper is on the premise that all simulation objects already distributed on the web.

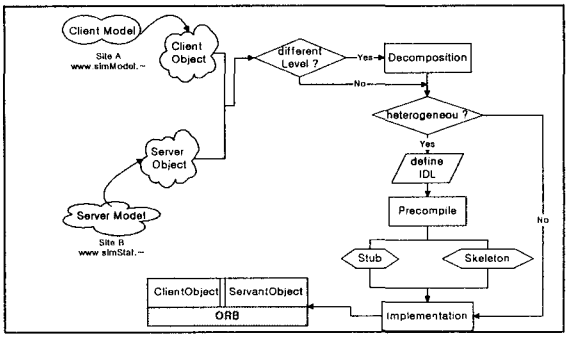

Figure 6. the whole architecture of integration of distributed simulation objects

On the different Internet sites, two simulation model are existed respectively. The procedure of integration of distributed simulation models steps the followings.

Step 1. It is necessary of each simulation model that be transformed into object. one object could be server object or client object relatively to the role of the other object. That is, a object would be server object when its role is mainly to receive and treat the requested messages, and a object would be client object when its role is mainly to send messages and request some results in the relationship between two objects. So one could have not simulation models but simulation objects.

Step 2. Then objects are compared whether they have the same level in terms of composition. If not, the object with the higher level should be made a decomposition into the object with the lower level. Decomposition approach is selected because it is easier than composition approach in the object-oriented paradigm and it is more proper in case of process of integrating objects. If two objects, which are located on the different sites and represent simulation models, have the same level, no need to decompose them into any levels. Now one has the simulation objects with the same level.

Step 3. The investigation, which means that two simulation objects are based on the different programming languages or not, should be done. If two objects are based on the different programming languages such that one is made of C++ and the other is made of Java, so called two objects are 'heterogeneous', the interfaces between two objects are defined by IDL in CORBA. IDL gives a few great benefits that are no need to modify the source code of each object, so called 'legacy', and the ease of writing interfaces because of the syntax simplicity of IDL itself. Then two network communication codes are generated through pre-compilation of defined interfaces. They are stub for client-side object and skeleton for client-side object. In these stub and skeleton the codes for being compatible with CORBA's ORBs.

Step 4. The next step is to write the detail implementations of server-side object, so called 'servant object' or 'object implementation' in CORBA, and client-side object. These implem -entation is what the client and server object do actually and one do not need to too much concern how the objects to communicate with each other on the network, especially Internet because ORBs cope with most of communica -tion issues between the objects.

5. EXPERIMENT

There are some cases when one tries to integrate distributed objects. The first of them is that one object is designed with the architecture of CORBA and the other one is

design with the architecture of non-CORBA. The second is that each object is built using different language. In addition to that, each object may be based on different computer languages. The simple experimental model is two node network model (See the Figure 8). Node 1 exists on the server-side computer and node 2 based on the client computer.

First of all users should define an interface between server/side and client/side node. It could be done by CORBA IDL. It is a very simple way. As described in advance, one should define module, interface, attribute, and operation.

```
module EventPool
{
    interface Event
    {
        attribute double timeStamp;
        double getNextEvent();
    };
};
```

One can find out that strategy pattern is used. Strategy pattern is one of behavioral patterns which are used to organize, manage, and combine behavior. Strategy pattern encapsulates related algorithms in classes that are subclasses of a common superclass. This allows the selection of algorithm to object and also allows it over time. Figure 7 shows what strategy pattern is in UML.

In this test, we first run the client in the machine-Pentium(MMX) with 200MHz processors and 64MB of RAM running Windows98, the server in the machine-Pentium II with 500MHz processors and 256MB of RAM running NT4.0. Then we run them across a 10.0Mbps LAN.

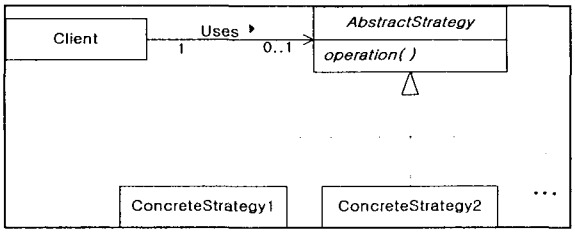

Figure 7: Strategy pattern (source: Grand, 1998)

5.1 CORBA with JAVA remote vs local

Two nodes are built with same computer languages. Java is used for servant class and client class. The premise of this prototype is all compatible with CORBA. Figure 8 shows how to generate codes for client/server nodes. This interface will be mapped to each language. IDL is precompiled with 'idltojava' precompiler. For generating distributed objects, server-side class or servant class is constructed based on _EventImplBase.java, Event.java, and _example_Event.java.

Especially _example_Event file, which is provided by Inprise's Visigenic VisiBroker for java, make a model builder convenient because it contains brief form of object implementation.

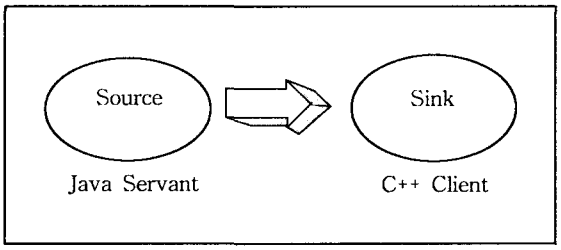

Figure 8. Two node network model

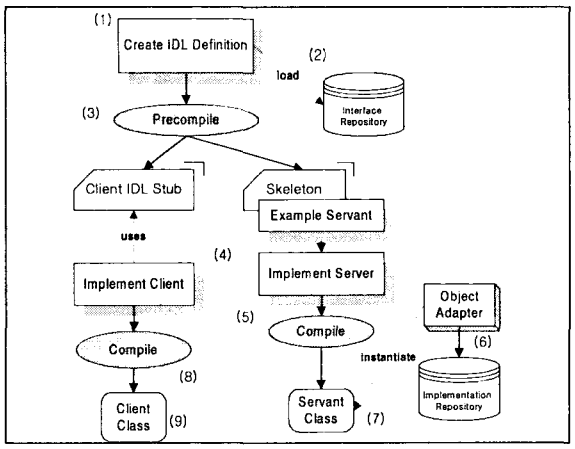

Figure 9. Procedure of generating client/server codes

5.1.1 CORBA remote

Code 1: EventClient

```
import java.util.*;

class EventClient
{
  public static void main(String args[])
  {
      try
      {
  //Initializing The ORB
        org.omg.CORBA.ORB orb =
            org.omg.CORBA.ORB.init(args, null);
  //Binding to Ramdon Variate
        RandVariate.Rand random =
      RandVariate.RandHelper.bind(orb,"EventPool");
        random.ramda((int)0);
        for(int i=0; i<1000; i++)
        {
           System.out.println("ramdom number = "+
                              random.getRnd());
        }
      }
      catch(org.omg.CORBA.SystemException e)
      {
      }
  }
}
```

Code 2: EventServer

```
class RandServer
{
   static public void main(String[] args)
   {
      try
      {
  // initializing ORB
        org.omg.CORBA.ORB orb =
            org.omg.CORBA.ORB.init(args, null);
  // binding the BOA
        org.omg.CORBA.BOA boa = orb.BOA_init();
  // creat the expon object
        RandImpl expon =
                        new RandImpl("EventPool");
  // export to ORB the newly created object
        boa.obj_is_ready(expon);
  // ready to service requests
        boa.impl_is_ready();
      }
      catch(org.omg.CORBA.SystemException e)
      {
      }
  }
}
```

The 'event' object is a part of actual implementation of servant class. And servant class prepares the ORB by calling init(args, null) method and initialize BOA which is Visibroker ORB. Thus servant object can set up all preparations and wait for client request.

5.1.2 CORBA local

In this test we first run the client and server programs in the same machine

5.2 C++ vs JAVA with CORBA

Two nodes are built with different computer languages. Java is used for servant class and C++ for the client class. Figure 10 shows structure for the interface.

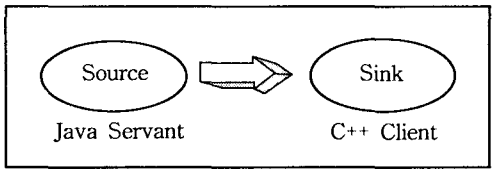

Figure 10. Two node network model

This interface will be mapped to each language. IDL is precompiled with 'idltojava' or 'idltocpp' precompiler. Then one can get the following files in Figure 11. Sequence is similar to test 5.1

On the other hand, Client-side class is constructed based on the event_c.hh and event_c.cpp.

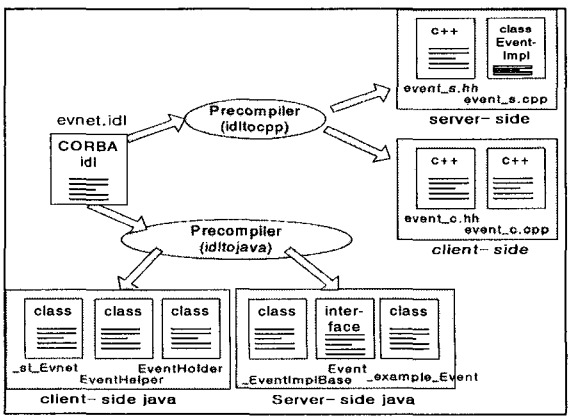

Figure 11. generated files after precompiling idl file

5.2.1 C++

Code 1: EventClient

```
#include "event_c.hh"
int main(int argc, char *const *argv) {
      try {
            // initializing ORB.
            CORBA::ORB_ptr orb
                  = ORBA::ORB_init(argc,argv);
            // binding to Event Object.
            EventPool::Event_var eventCalc =

EventPool::Event::_bind("EventPool");
            // set timeStamp to initial value of 0.
            eventCalc->timeStamp(0.0);
            // call 1000 times.
            double sum = 0;
            for (int i = 0; i < 1000; i++) {
            sum += eventCalc->getNextEvent();
            }
            .....
```

Client class include the 'event_c.hh' Therefore it can use implementation in CORBA. One can see that a client object called 'eventCalc' is bound with name of 'EventPool' and it can obtain an object reference. This name 'EventPool' is used in Naming service in CORBA.

5.2.2 JAVA

Code 2: EventServer

```
import org.omg.CORBA.*;
Class EventServer {
    static public void main(String[ ] args) {
      try
      {
            // initializing ORB
            CORBA::ORB orb
                = CORBA::ORB_init(args, null);
            // binding the BOA
            BOA boa = orb.BOA_init( );
            // creat the Event object
            EventImpl event =
                        new EventImpl("EventPool");
            // export to ORB the newly created object
            boa.obj_is_ready(event);
            // ready to service requests
            boa.impl_is_ready( );
        } catch (org.omg.CORBA.SystemException & e)
        {
        }
    }
}
```

Figure 12 shows relationship concerned about created objects.

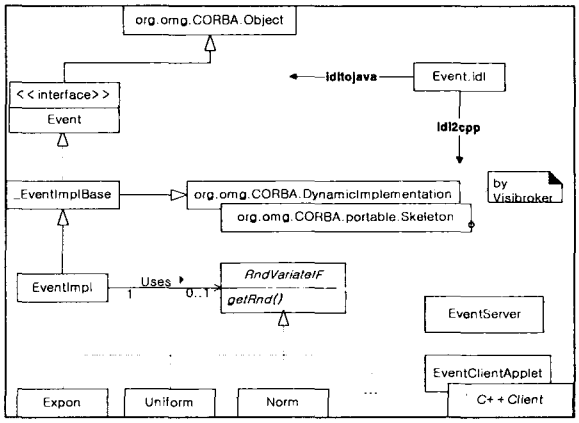

Figure 12: relationship between objects represented by UML

Executing this model, source node generates values of random variables having certain distributions such as exponential distributions, normal distributions and etc.send to sink node. Sink node invoked getRnd() or getNextEvent() method 1000 times, and then this process is repeated 100 times. Sink node conducts a simple calculation that checks the its mean value using data from source node. Table1 shows the comparison of experimental results explained above.

Table 1 : Comparative result

	CORBA with JAVA remote vs local		C++ vs JAVA with CORBA remote object invocation
	remote object invocation	local object invocation	
$\hat{\chi}$	2.4311 msecs	4.7325 msecs	4.7312 msecs

msecs - milliseconds

In general, remote object invocation are slightly faster than local object invocation.(See Figure 13(c)) The reason is which the program runs on two machines instead of one. However, it does tell us that the Java

Virtual machine is very CPU-intensive. C++ (client) vs JAVA(server)

with CORBA test will demonstrate that C++ ORB on the client side can talk to Java ORB on the server side via IIOP. Also the test demonstrates that allow invoke object between different languages.(See Figure 13).

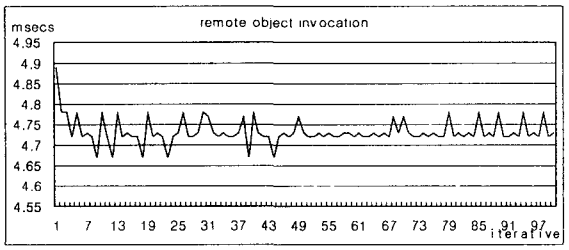

Figure 13 (a): Remote object invocation

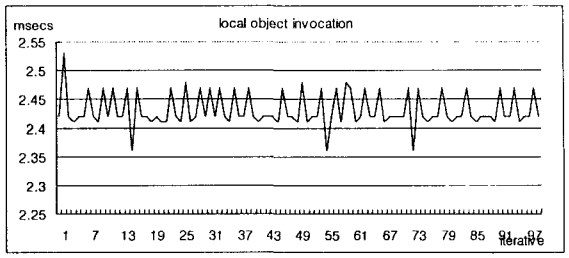

Figure 13 (b): Remote object invocation

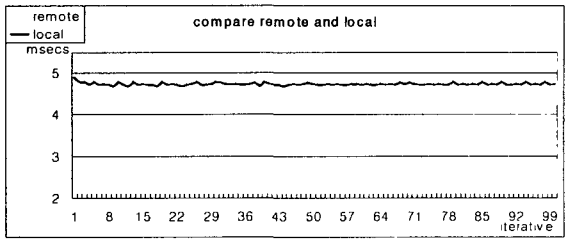

Figure 13 (c): Compare remote with local

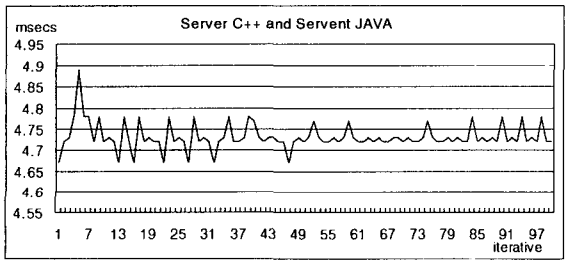

Figure 13 (e): C++ and Java

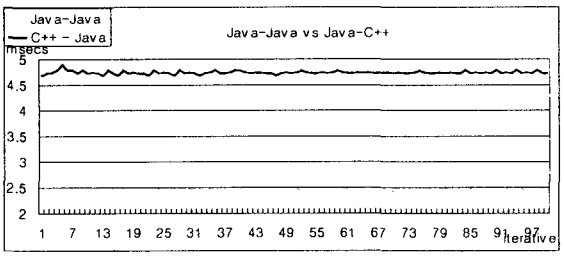

Figure 13 (d): Compare Java-Java with Java-C++

6. Conclusion

Model developer reduces modeling time, and risk while producing information needed for design, analysis and operation of complex production systems. It is quite necessary of a means that provides easily creating comprehensive system representations from reusable models rather than starting a new each time.

In this paper an architecture for dealing with simulation objects distributed on the web to build a simulation model is presented. Hence this research focuses on showing how to construct the simulation model by integrating the distributed objects on the web. To cope with the issues of distributed objects, web-oriented technolo- gies such as Java IDL, CORBA, and design patterns are used. Java IDL gives a great convenience to one who is using Java for building simulation model and wants to take advantages of CORBA. CORBA plays great roles of object integration as a middle-ware between heterogeneous simulation object. Thus CORBA enables the integration of simulation models that are vendor-neutral, language-neutral, and platform-neutral under the internet environment. One could achieve the extended reusability of simulation objects and common understanding about the constructed simulation models by using design patterns.

This research tries to develop the following major tasks. One is to developing a standard methodology to transform individual distributed simulation models into objects worth full network communication abilities. Another one is to define an environment and a standard architecture that brings together the distributed simulation model. The other is to present mechanisms for constructing and controlling simulation model networks on the web and prove capability of simulation model integration on the web. For this, many design patterns are introduced into simulation modeling works and a simple two node network model as an example was shown on the premise that all the distributed objects are originally composed in a CORBA -compatible way at the end of this paper.

Design patterns adopted in this paper are very helpful for someone who tries to develop simulation library and to figure out how the object integration performs. Another paper could be written on the further study about the following two tasks. One is to create intelligent simulation agents whose role are searching pieces of simulation sub-models, assisting a model builder and monitoring to conduct integrating distributed simulation objects, controlling implementation of simulation on the web. Effective search algorithm and monitoring function will be researched. For this, an effective formalization process of simulation model information should be preceded. The other issue is decomposition. To make each object having the same level, one object could be divided to match each other. An effective method should be considered to decompose and make the same level for integrating objects. For this, studying and finding out which design patterns are more appropriate to solve and define the overall design of a simulation model through decomposition and integration of distributed simulation objects.

References

[1] Banks, J., Carson, J. S. II, Nelson, B. L., "3. General principles", "4. Programming Languages", In Discrete-Event System Simulation, 2nd Ed., 1996, Prentice Hall, pp. 59-152.

[2] Buss, A. and Jackson, L., 1998, "Distri -buted simulation modeling: A comparison of HLA, CORBA, and RMI", In *Procee -dings of the 1998 Winter Simulation Conference*, pp. 819-825.

[3] Chan, A. and Spracklen, T., 1999, "Web -based distributed object simulation frame -work", In *Proceedings of the 1999 summer computer simulation conference*, July 11-15, 1999, Chicago, Illinois, pp. 9-14

[4] Eriksson, H. and Penker, M, 1998, UML Toolkit, John Wiley & Sons

[5] Fishwick, P.A. "Web-based simulation: Some Personal Observations.", Proceedings of the 1996 winter simulation conference, pp. 772-779, 1996.

[6] Fishwick, P. A., 1997, "Web-based simulation", In *Proceedings of the 1997 Winter Simulation Conference*, December 7-10, 1997, Atlanta, Georgia, pp. 100-102

[7] Fishwick, P. A., 1998, "Issues with Web-Publishable Digital Objects", *SPIE Aerosense Conference*, April 1998, Orlando, Florida

[8] Fishwick, P. A., 1998, "An architectural design for digital objects", in Proceedings of the 1998 Winter Simulation Conference, December 13-16, 1998, Washington, D.C.,

pp. 359-365.

[9] Gilbert, S. and McCarty, B., 1998, "Designing remote objects", "Designing persistent object", and "Architectures: Design-in-the-huge", In *Object-Oriented Design in Java*, The Waite Group, Inc., pp. 439-493, pp. 613-639

[10] Grand, M., *Patterns in Java*, vol.1, 1998 and vol. 2, 1999, Wiley

[11] Heim, J. A., 1997, "Integrating distributed simulation objects", In *Proceedings of the 1997 Winter Simulation Conference*, December 7-10, 1997, Atlanta, Georgia, pp. 532-538

[12] Joines, J. A. and Roberts, S. D., 1998, "Fundamentals of object-oriented simulation", in Proceedings of the 1998 Winter Simula -tion Conference, December 13-16, 1998, Washington, D. C., pp. 141-149.

[13] Joines, J. A. and Roberts. S. D., 1998. "Object-Oriented Simulation", In *Handbook of Simulation*, Edited by Jerry Banks, John Wiley & Sons, Inc. 397-427.

[14] Page, E., Buss, A., Fishwick, P., Healy, K., Nance, R. and Paul, R., 1999, "Web -Based Simulation: Revolution or Evolution?", *ACM Transactions on Modeling and Computer Simulation*, February 1999.

[15] Praehofer, H., Sametinger, J., Stritzinger, A., 1999, "Discrete event simulation using the javabeans component model", in Proceedings of the 1999 International Conference on Web-Based Modeling and Simulation, January 17-20, 1999, San Francisco, California, pp. 107-112.

[16] Schmidt, A., D pmeier, C., Eggert, H., 1999, "Wildflow - Components for building scientific and technical work environments", in Proceedings of the 1999 International Conference on Web-Based Modeling and Simulation, January 17-20, 1999, San Francisco, California, pp. 191-198.

[17] JavaTM IDL, in Java TM 2 SDK, *Standard Edition Documentation, Version 1.2.2*,

● 저자소개 ●

이영해

1977년 고려대학교 산업공학, 학사
1983년 Univ. of Illinois, 산업시스템공학, 석사
1986년 Univ. of Illinois, 산업공학 및 경영과학, 박사
1986년 ~ 현재 한양대학교 산업공학과 교수
관심분야 : Web-based Simulation, Simulation Output Analysis, Simulation Optimization

김숙한

1985년 육군사관학교 졸업
1991년 미 해군대학원 운영분석, 석사
현 재 한양대학교 산업공학과 박사과정
관심분야 : 시뮬레이션, SCM, ILS

심원보

2000년 한양대학교 산업공학과 석사
현 재 한국오라클 consultant
관심분야 : 시뮬레이션, SCM, ILS

김서진

2000년 한양대학교 산업공학과 학사
현 재 쌍용정보통신
관심분야 : 시뮬레이션, SCM, e-business