한국시뮬레이션학회 '99추계학술대회 논문집 1999.10.9 한양대학교

웹상에 분산되어 있는 시뮬레이션 객체들의 통합에 의한 시뮬레이션 모델링 방법론

심원보*, 이영해

한양대학교 산업공학과

경기도 안산시 사1동 1271번지

Tel.: 0345-400-5269, Fax.: 0345-409-2423, E-mail*: wbshim@pis.hanyang.ac.kr

A Simulation Modeling Methodology by Integrating Distributed Simulation Objects on the Web

Wonbo Shim, Young Hae Lee

Department of Industrial Engineering

Hanyang University

Abstract

Web-based simulation is one of the most interesting field of simulation research today. Among many research area of web-based simulation, we concern about what a effective way of building simulation model is since creating comprehensive simulation models can be expensive and time consuming. So this paper discusses how to integrate distributed simulation sub-models as objects for constructing the required simulation model which is more large and complex. We introduce two web-oriented methodologies (such as JIDL, CORBA) and the concepts of agent for assisting modelers to integrate simulation models scattered over the web.
SINDBAD, which we designed, is a simulation environment which makes it possible constructing a simulation model with distributed model objects on the web and performing the parallel simulation in a distributed way. It is organized according to design patterns in the object oriented concept. Actually we are on the premise that all the distributed objects are originally composed in a CORBA-compatible way to start with our prototype of SINDBAD.

1 Introduction

Recently the World Wide Web definitely represents a fertile area in which to perform computer simulation research. There are several directions that one can take in establishing a connection between the web and the simulation field. Two directions involve 1) parallel and distributed model execution, and 2) distributed model repositories (Fishwick 1997). As Fishwick did, we do focus more on the area of distributed model repositories since there has been less research in the area than in the more mature field of PDES 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.
By Fishwick, the concept of Digital Object has been introduced. One of the most critical problem in the field of computer simulation today is the lack of published models and physical objects within a medium - such as the WWW - allowing such distribution. 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, and 3) the VLSI Hardware Definition Language (VHDL).
In this paper, we concern how one can treat with probably heterogeneous and not standardized objects which are distributed on the web and how one can integrate them for simulating integrated models together.

2 Foundations of Dealing with Distributed Objects on the Web

2.1 CORBA

CORBA (Common Object Request Broker Architecture) is the standard distributed object architecture developed by the Object Management Group (OMG) consortium. Since 1989 the mission of the OMG has been the specification of an architecture for an open software bus, or Object Request Broker (ORB), on which object components written by different vendors can inter-operate across networks and operating systems. 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, and 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 below shows a method request sent from a client to a CORBA object implementation in a server.

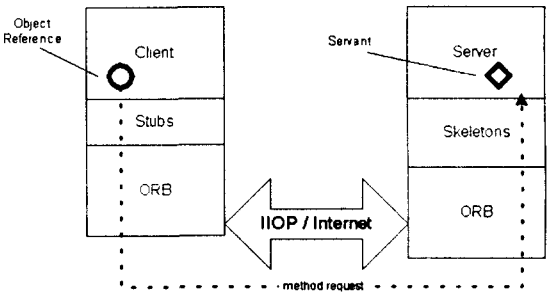

Figure 1: Implementation between client and server

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)

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.
No interface repository is provided as part of Java IDL. An interface repository is not required because under normal circumstances, clients have access to generated stub files.

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

Our basic approach for constructing a environment integrating objects ,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.
For example, Source node, which is derived from Node class, can be established in the following pattern so called Strategy. For representation of the design patterns, we use the Unified Modeling Language (UML).

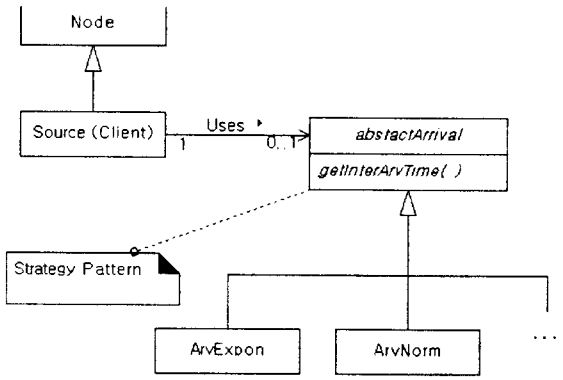

Figure 2: An example of Strategy pattern

3 Architecture of Integrating Distributed Simulation Models as Objects

3.1 Multi-tiered application

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, SINDBAD, which we named, using distributed objects is made up of three-tiered architecture, is CORBA -compatible application environment, 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 below illustrates the three-tiered, modular applications.

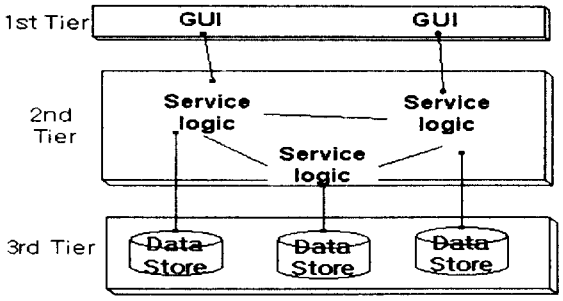

Figure 3: the 3-tiered application

3.2 Integrating distributed simulation

objects

simulation models as object

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. The functionality would be that set of activities the object would perform if requested, and the state of the object would be the value of all variables describing that particular object.

building a simulation model

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 try 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. And secondly we borrow us the concept of plug and play. As if we played with Lego blocks, we may bring to assembly the models by providing middle-ware as capsule of the objects. For this, we are on the premise that all the distributed objects are originally composed in a CORBA-compatible way to start with.

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 message-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.

agents

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.

The agent is created and maintained separately from the model. So sometimes the simulation server may prepare several agents which play the different roles each other to meet various requirements of simulation modelers.

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

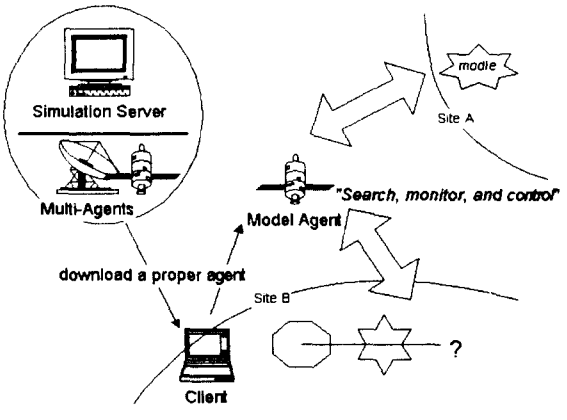

Figure 4: Interaction between simulation modeler and model agent

3.3 Algorithm of integrating objects for the agent

Agents are software programs that execute specific tasks on behalf of another party. 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.

We propose to apply a data mining technique for the intellectual agent, especially the algorithm of market basket analysis based on the association rule, case-based reasoning, and decision tree. 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. Instead, one can be rely on the possibilities that the search engines such as Yahoo, Excite, AltaVista, etc. will have a capability of finding simulation models as objects. So one browses the simulation models with the web browser such as netscape or internet explorer on the World Wide Web.

4 SINDBAD: An Environment 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 with 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 environment of SINDBAD.

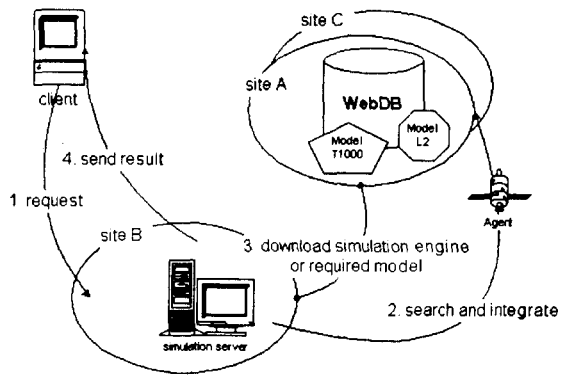

Figure 5: How to implement integrating for constructing simulation model and to perform simulation.

The Internet itself, holds a massive parallel processing power in building simulation models and performing simulation itself. We are still under the experiments that test the capabilities of parallel distributed simulation using SINDBAD.

5 Summary

Modeling reduced time, cost, and risk while producing information needed for design, analysis and operation of complex production systems. However, the costs of modeling, the expertise required, and the pains of starting anew each time are impediments to more wide spread adoption of simulation technology.
In this paper, we presented a architecture for dealing with distributed simulation models as objects. This paper shows how to construct the simulation model by integrating the distributed objects on the web.
Our development effort has been divided into the following major tasks:

1) Developing a standard methodology to transform individual distributed simulation models into objects worth full network communication abilities,
2) Creating agents whose role are searching, assisting (monitoring) to integrate distributed simulation sub-models as object, and controlling implementation of simulation on the web,
3) And developing an environment that brings together the tools and mechanisms to construct model networks, to monitor and control their interaction in performing a

simulation.

Java IDL gives a great convenience to one who wants to take advantages of CORBA. We could achieve the extended reusability of simulation objects and common understanding about the constructed simulation models by using design patterns.

Acknowledgement

The work presented here was supported by Institute of Information Technology Assessment in Korea.

References

Fishwick, P. A., 1998, "Issues with Web-Publishable Digital Objects", SPIE Aerosense Conference, April 1998, Orlando, Florida, http://www.cise.ufl.edu/~fishwick/tr/chron.html

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

Java™ IDL, in Java TM 2 SDK, Standard Edition Documentation, Ver. 1.2.2, http://java.sun.com/products/jdk/1.2/docs/guide/idl/index.html

Grand, M., Patterns in Java, vol. 1 (1998) and vol. 2 (1999), Wiley

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.

Buss, A.; L. Jackson.1998. "Distributed simulation modeling: A comparison of HLA, CORBA, and RMI", In Proceedings of the 1998 Winter Simulation Conference, 819-825.

Chan, A. and Spracklen, T., 1999, "Web-based distributed object simulation framework", in The proceedings of the 1999 summer computer simulation conference, July 11-15, 1999, Chicago, illinois, pp. 9-14

Gilbert, S. and McCarty, B., 1998, "Designing remote objects", "Designing persistent objects: Database design and Implementation", and "Architectures: Design-in-the-huge", in Object-Oriented Design in Java, The Waite Group, Inc., pp. 439-465, pp. 467-493, pp. 613-639

Page, E., Buss A., Fishwick, P. A., Healy, K., Nance, R. and Paul, R., 1999, "Web-Based Simulation: Revolution or Evolution", ACM Transactions on Modeling and Computer Simulation, February 1999, http://www.cise.ufl.edu/~fishwick/tr/chron.html

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.

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

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