ARCHITECTURAL RESEARCH, Vol. 7, NO. 2 (2005), pp. 23-33

Distributed Design System as a New Paradigm Towards Future Collaborative Architectural Design Process

Seung Hoon Han
Convergence Laboratory, KT, Seoul, Korea

Abstract

The use of computers in architectural professions has grown with the power of easy data management, increased sophistication of stand-alone applications, inexpensive hardware, improved speed of processing, use of standard library and tools for communication and collaboration. Recently, there has been a growing interest in distributed CAAD (Computer-Aided Architectural Design) integration due to the needs of direct collaboration among project participants in different locations, and Internet is becoming the optimal tool for collaboration among participants in architectural design and construction projects. The aim of this research is to provide a new paradigm for a CAAD system by combining research on integrated CAAD applications with recent collaboration technologies. To accomplish this research objective, interactive three-dimensional (3D) design tools and applications running on the Web have been developed for an Internet-based distributed CAAD application system, specifically designed to meet the requirements of the architectural design process. To this end, two different scopes of implementation are evaluated: first, global architecture and the functionality of a distributed CAAD system; and, second, the association of an architectural application to the system.

Keywords: Distributed Design System, Collaborative Design, Design Process, Distributed Collaboration, Internet

1. INTRODUCTION

Until recently, the use of computers in the profession of architecture has grown with the power of easy data management, updates, use of standard library and tools for communication and collaboration. The computing resources of an organization or project team are spread across many different platforms in different locations. This state of affairs is creating a growing interest in distributed CAAD integration due to the needs of direct collaboration among project participants in different locations. The potential for the integration of information is expected to have a tremendous impact on architecture and on the construction industry.

Internet is becoming the optimal tool for collaboration among participants in architectural design and construction projects because of the low connection costs and wide availability. Such collaborations will include the exchange of project drawings and various forms of project materials and general distribution of project information through the Internet. One way or another, the existence of the Internet and the wealth of related technology will change the way architectural design and construction are practiced today.

Distributed object computing has the potential to change the information landscape of a broad range of business practices. As integrated computer systems offer the capability to improve the effectiveness and efficiency of management processes in practice, their use is likely to increase the information flow and the quality of communication among project participants in the collaborative design process.

A typical large-scale architectural project, normally, involves participants in various disciplines, generating large volumes of data and decisions. Centralizing such large amounts of data in a single database poses technical difficulties. Distribution technology, however, can solve these problems by accomplishing the concept of network transparency, for example; data physically stored in many different locations can be seen as a single data repository.

This research will investigate how architects can make a successful collaboration using distributed technology, especially in the early design stages which mainly involve their cognitive work, and aims to gain insight into the advantages and shortcomings of such an approach. Through this paper, the architecture of a distributed collaborative architectural design system is investigated and some experiments that examine design workflow tasks performed within the environment are presented.

2. KEY CHARACTERISTICS OF DISTRIBUTED DESIGN SYSTEM

Distributed-object technology is considered to be the most flexible server-client system available. It encapsulates data and task logic in objects designed to run anywhere on networks, and to run on different platforms. These objects talk to existing applications by way of object wrappers, and manage themselves and their resources. Accordingly, distributed objects supply a paradigm for building universal, transparent and adaptive information infrastructure systems. This new computing paradigm, distributed object computing, is a blending of the cognitive and semantic integrity of objects with the distribution architecture of client/server technologies.

As distributed CAAD environments are a special case of distributed systems, it is reasonable to exploit features and services of general architectures for distributed systems. This section examines concepts and recent issues related to distributed computing as an evolutionary paradigm of the client/server system, analyzes available network-based collaboration solutions, and proposes a process model of a CAAD-integrated distributed design system for future industries.

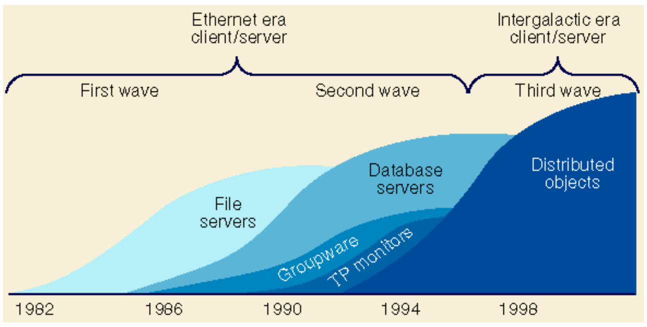


Figure 1. Three Waves of Client/Server Computing
(Source: http://www.byte.com/art/9504/img/412017c2.htm)

In the above figure, “Three Waves of Client/Server Computing,” the Ethernet era of client/server shows a file-centered application wave followed by a database-centered wave. As projected in the graph, distributed objects are the next big wave (Orfali et al., 1997).

One of strengths of distributed-object technology is that it can comprise other forms of client/server computing, including SQL databases, and groupware. Also, these distributed objects can help break large monolithic applications into more manageable components that coexist on the intergalactic network. In addition, existing systems can be “wrapped” and appear to the developer to be objects. Once wrapped, the legacy code can participate in a distributed object environment. Wrapping is a technique of creating an Object-Oriented (OO) interface that can access specific functionality contained in one or more existing computer applications. Technologically, this object interaction is accomplished through a sophisticated messaging system that allows objects to request services of other objects regardless of the machines on which they physically reside (Orfali et al., 1997).

The efforts to reduce cognitive complexity are evident in many areas of the computer industry. Object-Oriented Programming (OOP) and methodology are modeled after real world objects to reduce the cognitive burden. With familiar concepts in life and real, object-like classifications, programmers now have more space for their intuitive thinking, while having more freedom to program complexity. In this manner, OO methods were applied to the design of better programming languages as an interface between the human programmer and machine code.

The design features for object systems are inheritance, encapsulation, and polymorphism. These three characteristics of objects provide the central design benefit of object reusability. Another factor in reusability is a communications model using a message-based architecture. Messaging is a communications paradigm - send and continue work until notified - which allows for an object to be available for further requests and activities instead of waiting for a server to return a result. With asynchronous messaging, the client can be notified when the results are ready instead of waiting for data. This benefit of asynchronous messaging is evident in a large network with many objects.

Most of the descriptions of operations (behavior) and data (attributes) of an OO model reside within the objects themselves. Here, the only way to use or manipulate an object is to send it a message. The hiding of internal information within objects is called encapsulation. To use an object, the programmer needs only to be aware of what operations it offers and which messages the object responds to. The advantage of encapsulation is that the implementation of objects can change or be extended while keeping the way the object is used by the rest of the system. The result is that changes tend to be local to an object and maintenance is simplified. Furthermore, as OO information systems are implemented and additional reusable objects become available, programming becomes more a matter of assembly rather than coding.

In an information system, many objects have similar characteristics such as information structure and behavior, in other words, procedures and methods. The concept of classes means order to the world of objects. Classes are templates used to define the data and methods of similar types of objects. An object created from a class is referred to as an instance, distinguishing the object from the mold from which it was created (the class). Some objects of the same general type may need specific characteristics added to the type. A mechanism, called inheritance, is provided to address specialization. As the name implies, inheritance is a feature that allows one class of objects to acquire some or all of its information structure and behavior from another class, rather than force the developer to define the structure or behavior over again. Hence, inheritance is a useful mechanism for reuse of objects.

Polymorphism is a Greek term meaning “many forms.” When applied to objects, polymorphism allows the developer to use one name for similar kinds of operations or functions. Rather than create unique names such as drawCircle, drawRectangle or drawSquare, a single method named draw may be used. The mechanism of the distinguishing classes, depending on the kind of object sending a message, launches the appropriate method such as draw a square. Polymorphism can eliminate the need for complex IF, ELSE and CASE structures and can enhance the use of inheritance concepts. Developers need not be concerned with the details of how other objects select their operations as objects have polymorphism mechanism inside (Fingar and Stikeleather, 1996).

With the impact of Information Technology, business and product cycle times are decreasing while the speed of business change is increasing. Management tries to streamline operations, reduce overhead and squeeze more out of production and sales channels in order to maximize shrinking margins.

Business applications of the future will need to be spread across multiple specialized platforms and will cooperate with other applications. To meet the demands of business, firms are investigating information systems based upon distributed object computing technologies and paradigms (Fingar and Stikeleather, 1996).

Business processes are essentially human phenomena. Real business modeling requires that we model the way work is actually accomplished, the ways things are. Business modeling captures the real business entities and operations, then translates them into object models. Object orientation was developed in the 1960s to provide the capability to build models that reflect real systems.

In this OO method, objects interact by passing messages to each other. These messages represent requests for information or services. The physical glue that ties the distributed objects together is an Object Request Broker (ORB). The ORB provides the means for objects to locate and activate other objects on a network, regardless of the processor or programming language used to develop them on either client or server side. Thus, the ORB is the middleware of distributed object computing that allows interoperability in heterogeneous networks of objects. ORBs were designed to provide a means for locating, activating and communicating with objects while hiding their implementation details from the developer (Pohl and Myers, 1994).

In a distributed object environment, an application supports a business process or task by combining necessary business objects. This component assembly is a major method for developing distributed object applications, which acts like an application located in a single place. In this environment, objects interact through a messaging system that allows them to request functions of other objects regardless of their physical locations.

The hiding of implementation details within objects is one of the key features of OO technology that allows the management of complexity in distributed computing. In a distributed object environment, the application developer does not need to consider what machine or programming language was used to implement the server objects. The user's view of an application consists of distributed objects that may be written in a wide scope of programming languages and platforms. For example, one program written in C++ and running on one machine, a Java program running on a mainframe, a Smalltalk object running inside one user's workstation, and an Excel spreadsheet running on a microcomputer may all be composed in a single application (Fingar and Stikeleather, 1996).

The objects appear to the user and developer as familiar business objects, not machines, networks and programming languages. Users and developers can think of those objects only in terms of familiar business objects, not in terms of the technology.

Distributed object computing is an extension of client/server technology. However, there is a difference in its working process and its implementation. With client/server, there is generally an application running on a client computer while another application runs on a server computer. These two applications communicate across a network and relay data to one another, usually via some middleware provided in the form of an Application Program Interface (API) or library function call.

A distributed application is made up of objects, just as any other OO application. However, the objects of a distributed object application are spread over and run on multiple computers throughout a network. In a sense, client/server is a narrow scope version of distributed object computing (Sariyildiz and Schwenck, 1996).

With this technology, objects can be distributed on different computers throughout a network, living within their own library outside of an application, and yet appearing as if they were local within the application. Several technical advantages result from a distributed object environment.

All communication between distributed objects occurs in the form of messages, just as local objects within an application communicate, rather than applying network interfaces to each existing system.

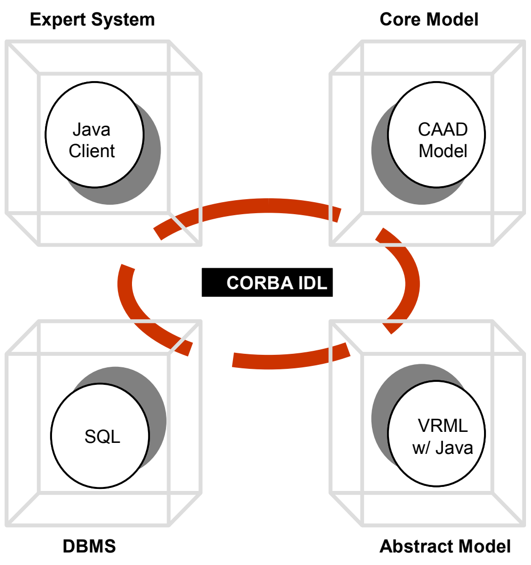
Since all objects - both local and remote - communicate in the same fashion via messages, programmers have the ability to distribute components of an application to computers that best fit the task of each object. For example, an object that performs intense computations, such as three-dimensional renderings, might be placed on a more powerful computer, rather than on an average desktop computer, where the user interacts with the presentation objects of the rendered images. This advantage can optimize hardware investments of an enterprise.

Software and hardware resources available on different platforms can be tied together into a single application. In this way, a single system image is achieved even when applications are assembled from distributed objects (Khemlani et al., 1998).

Two major standards for distributed objects are Microsoft's Object Linking and Embedding (OLE) and the Object Management Group's Common Object Request Broker Architecture (CORBA). In comparison with OLE, CORBA has more features for projects aimed at computer-integrated collaboration; it supports the principles of object-orientation in network computing whereas OLE does not.

The CORBA will be used as an advanced OO object connection provider in terms of system architecture concept and technology. As an integration technology, not a programming technology basically, CORBA will operate as the glue that binds different programming and systems together. Conceptually, it occupies the spaces between C++, Java and a DBMS (Database Management System) environment as a connection rather than a discrete component by itself. The expected power of distributed object computing will enhance the proposed system's performance with its unified system integrity.

The distributed object approach to integration has benefits when we consider the current technological and economic state of architectural collaboration. Instead of integration being achieved through static models that define the structure of shared information in the form of files or databases, the collaboration models can be distributed through a network to be easily accessed and modified from multiple users in different locations. This approach might, in particular, promote the use of computers on-site. Smaller, component-based applications are also easier to distribute over networks, particularly with the interpreted platform independent languages such as Java and C++ (Park 2001).



** IDL: Interface Definition Language*

Figure 2. An Example for CORBA Connectivity

3. PROPOSED SYSTEM ARCHITECTURE

The building design process has changed significantly in the last years. Generally, it is a matter of fact that the technological developments in every field of science have an influence on the society and therefore on the design and the design process itself. Architectural specialists are considering especially for the influence of the rapid developments of ICT (Information and Communication Technology) in architectural design (Sariyildiz et al., 1997).

The Internet has evolved as an excellent resource for the AEC (Architecture, Engineering, and Construction) disciplines, as it allows quick, efficient, and widespread communication to those who can access it, sharing everything from design information to project participant communication. Companies who previously marketed and sold CAAD products are now diversifying and offering services and other resources related to all aspects of the design industry.

Just as the earliest CAAD applications were relatively unsophisticated in their capabilities of making the drafting process of designers easier, these online services are currently in an early, formative period. The CAAD and AEC industries are relatively beginner on the Internet and so such services have strong as well as weak points. Already established CAAD companies enhanced the features of their offerings with innovation coupled with know-how, while newly joined enterprises to the industry realize of the needs of the industry viewed from new perspectives. Considering the consummate growth and widespread utilization by the AEC industry, the Internet and these Web-based services will be the greatest area of growth and development in the CAAD industry (Park, 2001).

Another new strategy for collaboration is proposed to empower designers in the architectural field with an innovative process, which comes from the utilization of distributed computing based on the OO approach. OO design applied to CAAD development lends favorably to the expected nature of distributed objects, which can considerably cut down decision-making procedures by providing cooperation between them; developments in CAAD technology has led to modular objects and eventually to their distribution. Distributed technology allows the designer to extract valuable information associated with the objects distributed online, not only values such as simple dimensions, but also other user-defined values from which reasonable updates and modification can be made.

Web services are appearing that cater to the AEC industry's need to collaborate efficiently and methods of implementing Web-enabled collaboration are arising. Recent peer-to-peer, distributed approaches are becoming a major trend of collaboration, although they have not been commercialized in the architectural profession yet. This approach provides a basis for all work to be done, concerning everything from project information to application without having to worry about obsolete or non-common hardware, software or unneeded personnel.

The current Web related programming technology, including Virtual Reality Modeling Language (VRML), ColdFusion, Java and Java Database Connectivity (JDBC), makes it now possible to implement a successful 3D information presentation system, which can be tested as a prototype model on the Web. With the understanding of architectural tasks and the specific nature of architectural data and communication, a 3D interface using VRML and Java can be designed to meet architectural design demands.

Concepts and tools such as Human-Computer Interaction (HCI), Data Exchange Standards, OOP and Web technology have all emerged from work on conceptual data models and network computing, and are apt to foster the development of a new paradigm that will enable researchers to take a new approach to CAAD. Indeed, the development of CAAD software applications, the development of new modeling methodologies and the definition of standards for information exchange create opportunities for achieving distributed system integration.

Therefore, the opportunity is seen to implement a solution which will provide both objects of basic usability to designers and the ready accessibility of those objects in the form of programmed applications over the Web, and will thus be manifested in the CAAD-enabled distributed system.

The proposed distributed CAAD system consists of the following major components: a database, a CAAD modeler, server application and interface. These components can be categorized by their residency. While the CAAD modeler and its project database reside with the architectural firms during the design phase, global project independent databases and expert applications are spread through the Internet environment to access the clients more easily. The framework of network connections among these components will be provided with the CORBA distributed object computing environment.

The database is defined by three levels of scale from large to small: global building database, local building database and project database. The global building database is built and managed by server database experts and covers a wide range of architectural data from simple drawing libraries to architecturally meaningful definitions such as room and wall. This database needs to follow a standardized data exchange agreement to be compatible with most local client firms. In a firm, the building database can be constructed by downloading relevant data from a global database server and is maintained locally. This building database should be small enough to concern only the firm's interest and large enough to avoid repetitive data downloading from the remote database server, maintaining efficiency. The project database is a subset of the building database and it is targeted to a specific project, which is currently in design development. This project database is usually integrated with the CAAD modeler in the design process.

Knowledge-based applications are available to the designer from remote domain locations through the Internet. These application servers get input from architects and give feedback either through an intermediary application viewer on the Web or directly back to the CAAD modeler through the Open Database Connectivity (ODBC). Along with the core model and the application viewer, a VRML model abstracted from the core model will be used to provide a better understanding of the project structure of the core model and the feedback from the remote applications. Hence, the future research of a core - abstract model mapping and manipulation tool is based on the importance of communication and interface design in a distributed CAAD system.

The core model is meant to be the center of our proposed distributed system. The core model is to be manipulated and maintained in the CAAD modeler using the modeler's built-in interface. This core model offers a complete geometric description of building project and gateway to the whole distributed system from a user's point of view when executing design tasks by generating and modifying the project model. In addition to the CAAD modeler's own interface utilities for direct manipulation, this core model will receive immediate feedback from the three other components by passing and updating information of current status to them. This is important in order to achieve direct manipulation among all four components in an integrated way, as well as individually. Thus, most of input and output operations will be performed through this core model to satisfy the issue of consistency and effectiveness.

The abstract model helps the architect conceptualized building data on various layers. The abstract model is empowered by direct manipulation both in its own environment and in the integrated environment of the core model. Integration is possible with direct mapping of information from the abstract model data to the core model data and vice versa.

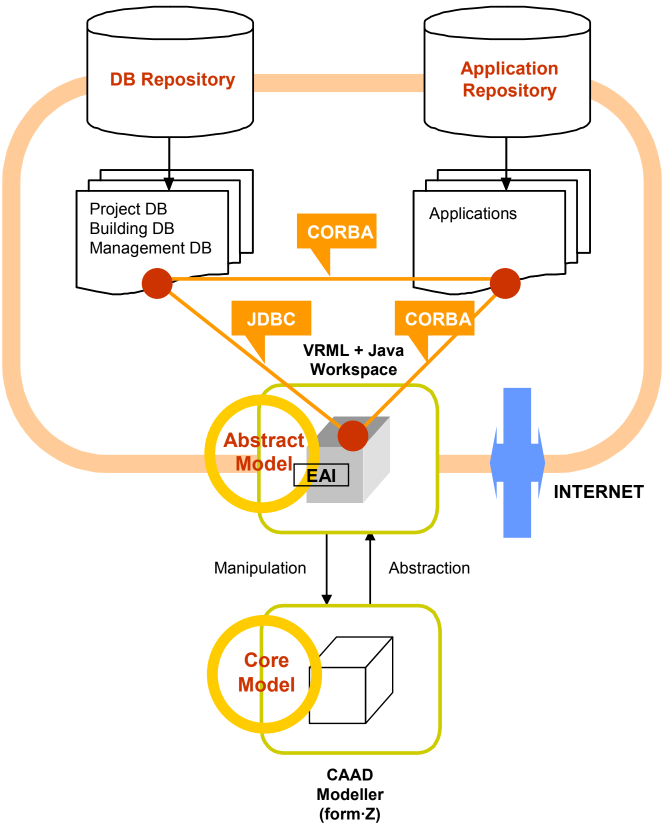


Figure 3. Proposed Distributed CAAD System

In architecture, methods like sections or walk-through are used to perceive abstract information on the traditional paper medium or through computer graphics. In more intelligent ways, the building model information can be abstracted as three-dimensional graphs. With nodes and path elements, the building model can be mapped into simplified skeletons of information directories. These directory structures are literally transparent. One can move around them. Hence, VR techniques of navigation can enhance one's ability to grasp the information structure in new ways (Park 2001).

The abstract model is designed to enable architects to browse building information interactively, in a hierarchical order. In the abstract model, building components can be classified with zones, floors, rooms, doors and windows, etc. These components are then assigned with simple 3D entities such as spheres and cubes based on their class. Selecting the 3D-node object explodes to the next level of a set of child 3D-nodes, which belong to the selected one. When selection reaches a 3D node with no child node, detailed information of the component is provided in a text format. Also, entity size and color give supplementary information about the corresponding building objects. When the mouse is left over an entity, additional information is given in a text window.

The proposed distributed system can be parceled into four discrete elements: a core model in the CAAD modeler, a supporting DBMS, an abstract visualization model in VRML browser and a knowledge-based application. A DBMS is involved in this system to support defining, constructing, and manipulating databases for other system components. All system components are connected to one another as part of a distributed CAAD system. The data flow in one or two directions: between 1) CAAD and database system, 2) VRML and database system, 3) CAAD and VRML application. The communication between components is supported by direct manipulation and direct information mapping. The issue of direct manipulation in part of the core model will be continued in future studies.

For Distributed Virtual Environments (DVE), objects have a graphical representation (scene-graph), an internal state and a behavior usually defined by program code. Such objects have to be added or removed from a scene in real time, their behavior has to be tracked in real time and their implementation has to be distributed immediately on different computers in the network. Some objects can be controlled by other objects and they should be able to share information.

Further requirements result from security considerations, e.g., protecting a scene from vandalism. An object can grant or deny access to its data and behaviors. For this purpose, it could provide user rights. In a process called authentication, the identity of users and their objects has to be checked.

When several objects try to access a shared resource, a conflict can arise. Such conflicts can be avoided through transaction mechanisms or they can be resolved by conflict solution strategies towards decision-making (Diehl, 2001). Other requirements are those common to distributed systems on the Internet.

1. Low Bandwidth and Network Latency: On the Internet, the bandwidth is low in general and there is no guaranteed bandwidth. Network latency is the amount of time it takes to deliver a message over the network. To hide network latency, a client can, for example, perform speculative computations or use buffering (Diehl, 2001).
2. Heterogeneity of Networks: Computers on the Internet run with different operating systems. These operating systems often differ in the programming languages and libraries they provide. A solution to the problem is to use platform-independent languages like Java and VRML, or architectures like CORBA, which achieve platform-independence through standardized protocols.
3. Distributed Interactions: Objects which interact in distributed systems can be controlled by programs or users at different computers. The computations that involve clients must be synchronized.
4. Scalability: A distributed system scales up if it works with large numbers of clients and objects. To achieve scalability, it is ideal that work is equally distributed to all clients and there are no bottlenecks in the communication structure. Most recent rendering techniques in VR have been developed on high-end graphics workstations. On the Internet, a scene needs to be rendered on computers with less computing power and thus rendering algorithms are preferable which scale down, i.e., yield near-photo-realistic results on high-end machines, but also less precise but acceptable results on personal computers.
5. Failure Handling: There is a trade-off between reliability and speed of transaction of messages on the Internet: lost messages must be resent. Monitoring the quality of transmission can be used to adapt it (Greenhalgh, 1999).

4. A WORKING PROTOTYPE

There can be various architectural applications available in a distributed virtual design environment. Designers and engineers can meet in the virtual counterpart of a new building before the first foundation stone is laid. In the early design stages, the effects of building design projects can also be visualized through a virtual world for participating decision makers. This way, collaborative design processes can be simulated without consumption of real materials. Nowadays, such experiments are getting more effective for the actual building process by using Distributed Multi-User Technologies (DMUTech) for the Internet.

This section is concerned with the design and realization of distributed collaborative virtual environment using DMUTech, named ARCH:DMUVR (ARCHitectural Support of Distributed Multi-User Virtual Reality), a working prototype of 3D computer-generated design environment, which actively supports collaboration between distributed participants. The approach taken in this system reflects both the management of interpersonal communication and the utilization of connection in distributed systems.

The role of ARCH:DMUVR here is to help architects make better design decisions with real-time presentation, communication, collaboration, feedback, and evaluation, especially in geometric aspects of the building design. The following key characteristics of the distributed systems have been reviewed and prototyped in terms of computation to gain successful implementation of ARCH:DMUVR:

1. Presentation: There is a notional world or space presenting design proposals, which is the virtual environment, generated from the core design model, activated as the abstract model, and visualized by VRML plug-in for the Internet browsers.
2. Representation: Every client is represented or embodied within the virtual environment using avatars and is visualized to other users by means of this embodiment to enhance access and comprehension. Each user is autonomous and able to move independently around the

virtual environment. Building design components which are objects in the virtual environment are extracted from the scene-graph, and represented by a separate Graphical User Interface (GUI) called 3D Building Object Editor for the system (*See Figure 6, left side*).

3. Communication and Collaboration: Participants can communicate and collaborate in many different ways through different communication channels within the computational and networking domains. A chatting window, for example, is provided for textual intercommunication using a specific communication channel, and a 3D Building Object Editor is used for distributed collaborative design among distributed participants through a different communication channel.
4. Negotiation and Decision-Making: The proposed system is also concerned with improving the support for collaborative decision-making based on observations of critical issues in agreement and negotiation drawn from the discipline of social science. When a participant in the design team attempts to update a design proposal, for instance, an agreement must be reached from all other connecting users via a networked agreement procedure in the system.
5. Evaluation: The most important concept for evaluation is observations of and reflections on the effectiveness and shortcomings of the distribution aspects of the developed system. A dynamic Web-based system evaluating application is implemented for the purpose of those observations, and its data are gotten from the user inputs, stored as weighted-values in the server, retrieved by the clients when requested, and reviewed by experts for future development.

The proposed prototype system supporting the above features is an integrated application environment, using DMUTech, which is accessible and usable to all the experts in the building design team, and which supports a wide spectrum of collaborative activities in the following three major realms of integration; ARCH:DMUVR aims to support not only the sharing of information [Data-Control Integration] but also the sharing of understanding [Control-Interface Integration] by providing the design developing tools for different aspects that can be plugged into it, and detail the additional solution to a shared building representation [Data-Interface Integration] for the virtual design environment (Han, 2005).

A VRML browser usually allows two primitive network operations: hyper-links and inclusion of media stored on different servers in the network. DMUTech is used for all aspects of network communication in multi-user worlds which have not been provided by the VRML browser. Essential requirements of DMUTechs are listed below (Roehl et al. 1997):

1. Adding, Removing, and Modifying Objects: If a user enters or leaves the world, or adds, removes, or alters an object, these changes must be performed in the view of all users. Users and objects must be registered; some objects might be owned by specific users.
2. Dispatching Changes: If an avatar or a scene object changes its position, orientation or state in some way, its new information must be the same in the views of all users. Users may have different rights to change objects and to choose modifying applications.
3. Text and Media Transmitting: Real-time text or media transmissions, similar to those used for Internet-conferencing tools, should ease communication among users.

The VRML browser and the Java applets can communicate via the EAI (External Authoring Interface). But, for the transmission of time-uncritical, large-scaled messages among browsers, it would be better to use CORBA and its network protocol, IIOP (Internet Inter-ORB Protocol). These browsers are now capable of communicating through an ORB. The programmer no longer needs to write code for sending messages to other hosts, but simply calls methods of the objects, which actually exist at other hosts where the methods get executed. For textual interaction between browsers, which is actually interacted by participants, IIOP is too slow, and TCP (Transfer Control Protocol) is more appropriate for their communication.

Java offers a rich set of protocols for network communication. These include HTTP (Hypertext Transfer Protocol) connection, TCP, and distributed objects using IIOP, but it also provides interfaces like the EAI to access Java programs. Hence, it is possible to control a 3D scene from Java or a Java applet. Other instances of the system can run on different computers and these instances can communicate, synchronize and enforce consistency by using Java network programming.

In sum, the following are used in the implementation of ARCH:DMUVR:

- Java- and EAI-based Approaches: With the EAI, it is possible for an applet in a standard Web browser to access the scene-graph of an embedded VRML browser as a Plug-In. Thus, a multi-user browser with integrated DMUTech can be implemented as Java applets in a Java-enabled browser. This approach requires communication channels using TCP, a wire protocol between application layers.
- CORBA-based Approaches: CORBA is an architecture for distributed objects in heterogeneous networks and allows objects to mutually access their services. The services provided by an object are specified as Interface Definition Language. These specifications are helpful not only for the programmer but also for other objects invoked dynamically. In the CORBA architecture, objects can be implemented in different languages. CORBA also

offers a variety of services for distributed systems (Han, 2004).

TCP establishes point-to-point connections between the server and the client. A number of messages can be sent over these connections in both directions. To communicate with a computer, a socket is created and then the input and output streams of the socket are accessed in a similar way for the connected URL (Uniform Resource Locator).

In contrast to TCP, IIOP delivers messages directly between peers without passing the central server. In distributed systems, when a request to create an object in remote memory is noticed, a surrogate referring to the remote object is also created. In the local system, since messages cannot be sent to the remote object directly, they are alternatively sent to the surrogate. The surrogate by itself cannot process the request, so the ORB gets involved to intercept and forward the message appropriately. The ORB on the remote system translates a service request from a request through the local language, such as Java, C++, or COBOL, to the implementation-neutral IDL, and forwards this request via IIOP to the ORB on a server system where a correct provider object is located. The server's ORB then translates an incoming request to the local language and forwards it to the repository to search for the provider object for processing. This way, different databases and applications can communicate as long as they conform to the CORBA standard, even if they are written in different local languages (Watanabe and Komatsu, 1997).

The protocol between clients and the server is specified by the IDL. An interface is a set of signatures which consist of the method name, its arguments and their types as well as the method's result type. It is possible for different DMUTechs to detect their distributed methods via CORBA's language-independent interface definitions and dynamic invocation.

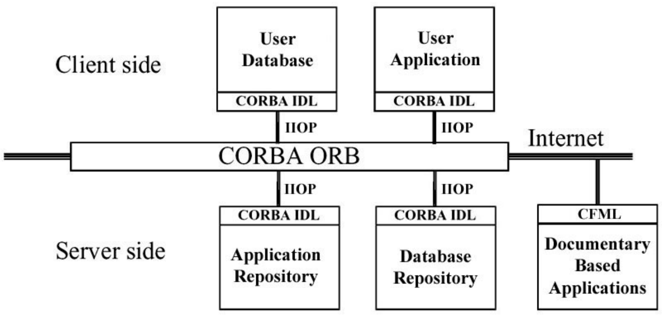


Figure 4. Communication Architecture of CORBA

Most geometric elements of the building design brought into this editor from the VRML browser lend themselves to implementation and presentation as a service in the system. Since there is diversity in realizing communication between different design processes, such as between different participants who have different knowledge-bases, the main communication option of the 3D Building Object Editor is direct peer-to-peer communication using CORBA. In addition, the central server which already controlled a few other communication channels cannot become a bottleneck in this way.

In summary, ARCH:DMUVR is a kind of hybrid architecture that handles the following six main communication channels relative to the key features of the system introduced previously, for effective collaboration on the network:

- Virtual Reality Presentation: TCP.
- Avatar and Object Representation: TCP.
- Communication Interface: TCP.
- Collaborative 3D Building Object Editor: IIOP.
- Agreement/Voting Interface for Negotiation: TCP.
- Evaluator: IIOP.

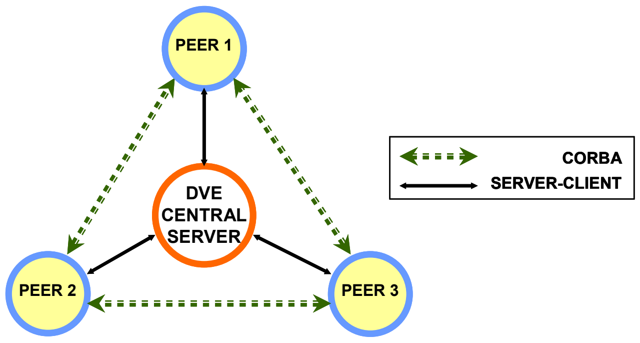
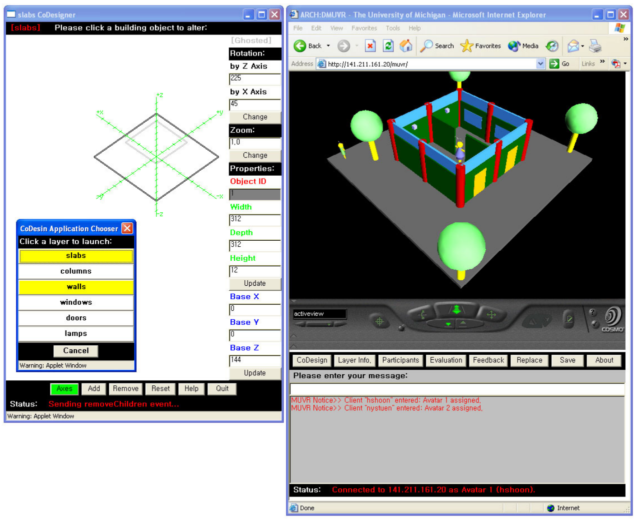


Figure 5. ARCH:DMUVR Connectivity

ARCH:DMUVR aims at providing a virtually integrated work space, although its contents are comprised of various components brought from many sources. The prototype can be started with Design Visualization Interface, the way in which ARCH:DMUVR is accessed by normal users and visualized to all connecting participants. It is anticipated that the system will normally be used with 3D graphics, and this is basically reflected in the design of a 3D GUI and has been supported by VRML browsers as shown in Figure 6, right side. At the top of this main window is the 3D graphical view of the building design visualized by CosmoPlayer. An abstract VRML model is launched into the interface for the previously selected building model, and clients' avatars are bound to the built VR environment. This shows a view into the virtual world as the graphical client of ARCH:DMUVR. At the bottom of the window is the communication interface which makes it possible for users to communicate with each other about the building model through textual exchange and share information. The user connecting to the system requires an authentication.

In multi-user environments, avatars play an important role as the virtual representation of a user. It is located at the viewpoint of the representing user from which she or he looks at the scene. The shape of the avatar determines how the user is seen by other users. If a user navigates through the scene and moves the viewpoint, the avatar also moves in the views of the other users.

Specific building component data to be used in the application server can be obtained either continuously from the client's local computer using middleware via IIOP, as long as the connection with the client is maintained, or discretely by packages of data through the database repository at reasonable time intervals. While connected to the server application, the client can receive real-time mapping results from the server.



```
225
45
1.0
1
312
312
12
0
0
144
http://141.211.161.20/muvr/
Status: Sending removeChildren event...
Warning: Applet Window
Please enter your message:
MUVR Notice>> Client "hshoon" entered: Avatar 1 assigned.
MUVR Notice>> Client "nystuen" entered: Avatar 2 assigned.
Status: Connected to 141.211.161.20 as Avatar 1 (hshoon).
```

Figure 6. 3D Object Editor and Design Visualization Interface

ARCH:DMUVR has a feature to bring data into the 3D Design Object Editor, which is a GUI support of the collaborative design environment; that is, CoDesign on the menu bar of the initial ARCH:DMUVR system calls the Application Launcher, which shows available applications distributed on the Internet and allows the user to launch those applications.

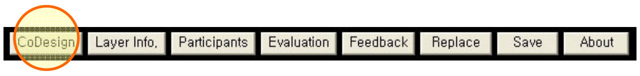


Figure 7. Menu Component of ARCH:DMUVR

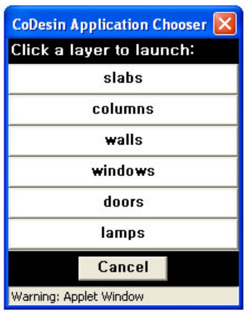


Figure 8. Application Launcher

Whenever an application is requested from the user by choosing an item from the Application Launcher, it searches the Internet using the naming service provided by CORBA, and connects to the local computer which owns relevant applications and data. All loaded geometric data from the local computer can be mapped to the 3D Design Object Editor, and transformed to 3D graphic components. Modified data using this editor are passed to the Design Visualization Interface, examined above, for a VR presentation, and dispatched to all participating users for collaboration.

3D geometric building data transferred from the client's local database are mapped to the 3D Design Object Editor and visualized in the Design Visualization Interface. The Design Object Editor is a specific application that is owned by a client and distributed on the Internet. Currently, ARCH:DMUVR can connect six different distributed applications and databases categorized by building components, such as slabs, columns, walls, windows, doors, and lighting; it is assumed that each application belongs to a different knowledge-based client. Those applications control the components of the scene-graph in the Design Visualization Interface, and linkage between the two interfaces is maintained by the application server, which is an owner of the application.

This 3D GUI is the way in which design proposals are authorized in 3D graphics. The building objects can be drawn using 3D graphic algorithms at the central canvas, and can be modified by altering the properties at the right-side in this interface. Altered data using this editor are transferred to, not only the central server to dispatch them to all clients for their modified VR visualization, but also to the application server from which the 3D Design Object Editor was originally launched, in order to pass those data to other clients' Design Object Editor interfaces and to change their drawings simultaneously. The current version of the 3D Design Object Editor supports collaboration only in geometric aspects of the building design due to limitation of the building property data.

Supplemental functions for design manipulation are also provided at the bottom of the interface. Such tools are not for collaboration, but for personal operations between current single user's 3D Object Editor and the VR Visualization Interface. Those actions are network-independent and not broadcast to other clients. For example, as shown in Figure 6, when a user wants to remove the upper slab from the scene for a better view, this action can be requested at the 3D Object Editor, and the results are shown directly in his visualization interface. The axes on the canvas can also be toggled individually.

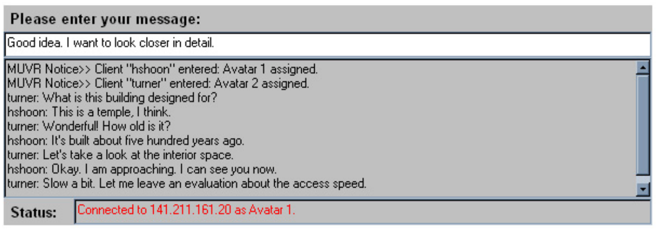
One of important requirements for operating ARCH:DMUVR is the security consideration. During collaboration, the scene-graph must be protected all the time from any possible bad behaviors or accidental changes. For this purpose, at the middle of the design update procedure is the Agreement Interface. This interface plays an important role in mediation of different opinions arising from the decision-making process. When a client attempts to update a design proposal with the 3D Object Editor, this action is immediately notified to all other clients who can grant or deny access to data and its behavior by responding to the Agreement Interface.

This interface uses the voting system, although the rule of getting agreement can be considered in various forms. Users' responses are anonymously collected in the server machine, and the server returns the voting result to all clients. Once the applicant for the design update acquires the votes for agreement from half or more of the respondents, the design proposal can be modified. This networked agreement procedure will help the design participants collaborate in a reasonable, positive manner (Han, 2004).

Every time the user navigates through the scene, the VRML browser sends position and rotation information to the server. When a second or third browser loads the same scene, it also connects to the server. From that point, movement information from one browser is sent to other browsers and is shared via the server. With this interaction between users, the VR system interface will make it possible to develop a real-time collaborative system for architectural environments via the Internet.

The ARCH:DMUVR system includes a communication interface which is a tool for sharing clients' messages, remarks, and immediate thoughts while navigating through the virtual places. All open interactions occurring are displayed in a textual chatting window identifying other clients and their actions within the scene. This communication interface is called when the client-side applet is activated and the contents of the communication are recorded in a server-side log file.

Afterwards, all the information logged into the system can be analyzed, and attached to design updates for tested environments. The communication interface has a link to the evaluation interface which enables the clients to leave more specific comments. This feedback procedure will enable every member-user to participate collaboratively in a step-by-step design process.


Please enter your message:

Good idea. I want to look closer in detail.

```
MUVR Notice>> Client "hshoon" entered: Avatar 1 assigned.
MUVR Notice>> Client "turner" entered: Avatar 2 assigned.
turner: What is this building designed for?
hshoon: This is a temple, I think.
turner: Wonderful! How old is it?
hshoon: It's built about five hundred years ago.
turner: Let's take a look at the interior space.
hshoon: Okay. I am approaching. I can see you now.
turner: Slow a bit. Let me leave an evaluation about the access speed.
```

Status: Connected to 141.211.161.20 as Avatar 1.

Figure 9. Communication Interface

The Design Evaluating System is to be integrated into the ARCH:DMUVR system as an evaluator of the modified environments as a result of collaboration. Two different aspects of design evaluation can be considered: One is a qualitative evaluation which is reflection and feedback including subjective thoughts of the design update in point of the beauty of the interior or exterior space. For this purpose, an interactive evaluation interface using dynamic Web forms through server-side ColdFusion scripts assist to create new evaluation values, enter clients' evaluations, and save them into the database with login information. This ColdFusion access to the database and to Java applications is possible by JDBC technology based on the ODBC. In general, ODBC enables direct connection, making two discrete systems integrated. JDBC serves as a mechanical joint between programming code objects and project database entities.

Evaluations consist of scoring evaluation criteria by selecting a radio button associated with a score value, and leaving a comment for each criterion in a text field. Scores and comments entered in the form are sent to the server-side and saved in the system database with a client ID and a date written. A later appendix from other clients can be filled out through the same interface and added immediately to the database as new records. Then, all evaluations stored will be retrieved and displayed on the result form in chronological order (Han, 2004).

The other evaluation is quantitative and relative to building performance for the modified design proposals. This kind of evaluation uses objective variables and rules for measuring building performance, and contains many arithmetic operations. Design Evaluating Applications for this purpose are built as distributed applications, which are located at and called from distributed experts' computer systems connected to the Internet. CORBA technology is applied to the implementation of this evaluator integrated in ARCH:DMUVR; The Design Evaluating Application remains ongoing and will be the subject of future work.

The role of the Design Evaluating Application in the system is to help architects make better design decisions with immediate feedback. It is proposed as a discrete application, which will run on a remote server and be executable on the Internet. This results from the assumption that in the future, bigger expert applications will be built on a server domain and will provide evaluation results to clients.

5. CONCLUSION AND DISCUSSION

This paper presented the possibility of a distributed system for architectural purpose. The distributed object approach to integration has benefits when we consider the current technological and economic state of architectural collaboration. Instead of integration being achieved through static models that define the structure of shared information, the collaboration models were able to be distributed through a network to be easily accessed and modified from multiple users in different locations. This approach will promote the use of computers on-site, in new terms of Network-Aided Architectural Design.

While experimenting and simulating ARCH:DMUVR in collaboration with various professions in the architectural field, the following have been observed:

- ARCH:DMUVR is a prototypical product of the distributed system model. As such I think it has been successful. It has provided a demonstration platform to explore the concepts of collaborative design.
- A framework of distributed object computing environments is usable, since it has been help-

ful for holding distributed meetings in diversity of professionals over networks.

- The current focus on peer-to-peer communication, based on sharing computation with other machines in the network, is promising. It has been particularly satisfying to see other experts deriving new insight from its use and enjoying involvement in the virtual design process with the system.

The contributions of this research fall into the following categories:

- The concept of a *core* model combined with the distributed computing approach articulates the complex nature of design development and information management into one well-defined domain. Using this core model as a central work object, the system can eliminate inconsistency of information storage and flow.
- The visualization and manipulation of the *abstract* model lightens the architects' cognitive burden in various design tasks. 3D-based direct manipulation and mapping concepts are incorporated in the design and implementation process. The potential of a Web-based modeling markup language such as VRML is demonstrated, and also the power of an OOP language is examined; Java offers satisfactory flexibility and availability of this abstract model to the distributed CAAD system.
- A knowledge-based application, directly connected to three other components (legacy modeller, abstract model, and DBMS) makes it possible to test the performance of the *distributed* system as a design tool; adaptability of the system for future uses with other applications are also evaluated.

The main theme around which this work has been organized is that of distribution. Existing integrative multi-user design systems can be used to build collaborative design authoring system; however, they do not explicitly consider distribution as a tool of communication and collaboration.

The suggested system attempts to become a general-purpose distributed VR system with features including a well-defined API; non-specialist world authoring tools; direct manipulation of virtual objects; and a systematic model of object behavior.

It lacks, however, support for popular graphical file formats of core models, such as AutoCAD and form·Z; that is, results of virtual object manipulation and world implementation with abstract models cannot be transmitted back to the core models in the current version. This feature will be researched and developed for the next version of ARCH:DMUVR. 3D-based direct manipulation and mapping concepts are also incorporated in the design and implementation process as future work. In addition, qualitative evaluation tools for the design alternatives generated by ARCH:DMUVR are designed to be built as distributed applications that will run on the Internet. CORBA technology will be utilized for the implementation of the evaluator that will be integrated in the next version of the ARCH:DMUVR system.

This research proposed a working prototype with an abstract model designed to provide a CAAD information interface for the Internet. This abstract model not only provides supplementary visualization tools for in-house project participants, but it also becomes an interface for remote participants who need a simplified project information-browsing tool. In sum, the Internet provides participants in the design team with a low cost collaboration environment.

REFERENCES

Diehl, S. (2001) "Distributed Virtual Worlds – Foundations and Implementation Techniques Using VRML, Java, and CORBA," Springer, Berlin, Germany.

Fingar, P. and Stikeleather, J. (1996) "Distributed objects for business," http://www.sun.com/sunworldonline-/swol-04-1996/swol-04-oobook.html.

Greenhalgh, C. (1999) "Large Scale Collaborative Virtual Environments," Springer, London, Great Britain.

Han, S. (2004) "A Working Prototype of Distributed Collaborative Architectural Design System," University of Michigan, Ann Arbor, Michigan, USA.

Han, S. (2005) "Thoughts and Tools of Collaborative Architectural Design Process," Architectural Research, Vol. 7, No. 1, Architectural Institute of Korea

Khemlani, L., Kalay, Y., and Choi, J. (1998) "Integrated Model to Support Distributed Collaborative Design of Buildings," Automation in Construction, Vol. 7, No. 1, Elsevier Science, New York, USA.

Orfali, R., Harkey, D., and Edwards, J. (1997) "CORBA, Java, and the Object Web," http://www.byte.com/art-/9710/sec6/art3.htm.

Park, H. (2001) "Distributed Representation of an Architectural Model," Graduate School of Design, Harvard University, Cambridge, Massachusetts, USA.

Pohl, J. and Myers, L. (1994) "A Distributed Cooperative Model for Architectural Design," Automation in Construction, Vol. 3, Elsevier Science, New York, USA.

Roehl, B., Couch, J., Reed-Ballreich, C., Rohaly, T., and Brown, G. (1997) "Late Night VRML 2.0 with Java," Ziff-Davis Press, Emeryville, Great Britain.

Sariyildiz, S. and Schwenck, M. (1996) "Integrated Support Systems for Architectural Design," Proceedings of the 3rd Conference on Design and Decision Support Systems in Architecture and Urban Planning, Belgium.

Sariyildiz, S., Volker, H., and Schwenck, M. (1997) "Improving CAAD by Applying Integrated Design Support Systmes and New Design Methodologies," CAAD Futures 1997, Kluwer Academic Publishers, The Netherlands.

Watanabe, S. and Komatsu, K. (1997) "The Distributed Architectural Model for Co-operative Design," CAAD Futures 1997, Kluwer Academic Publishers, The Netherlands.

(Data of Submission : 2005. 1.14)