A PROPOSAL FOR COLLABORATIVE VIRTUAL ENVIRONMENTS ARCHITECTURE USING INTERNET PATTERNS

Bianchi Serique Meiguins[1,2], Luis Affonso Guedes[1], Marcos Venícios Araújo[1,2], Marcelo de Brito Garcia[1,2] , Rosevaldo Dias de Souza Jr[1,2]

[1]Programa de Pós-Graduação - Universidade Federal do Pará (UFPA)
Departamento de Engenharia Elétrica, C.P. 8619 - CEP 66075-900 Belém - PA – Brasil.

[2]Núcleo de Pesquisa Tecnológica, Centro Universitário do Pará (CESUPA)
Av. Governador José Malcher, 1963, CEP 66060-230 - Belém – PA – Brasil.

bianchi.serique@terra.com.br, affonso@dca.ufrn.br, maraujo@cesupa.br

{mbgarcia, rosico}@amazon.com.br

ABSTRACT

Most of the problems related to Collaborative Virtual Environments (CVE) involve communication support, which should allow any update to the interface to be quickly noticed by other users. To minimize these difficulties, this paper proposes architecture for the communication support on CVE. The goal is to allow the developer not to worry so much about the communication layer during the development of the application. As basic requisites, the architecture should be easily extensible, updateable and support a variety of applications. The design of the architecture of the collaborative virtual environment is object-oriented and based on UML (Unified Modeling Language).

KEYWORDS

Collaborative Virtual Environments, Object-oriented Design, UML.

1 Introduction

Collaborative Virtual Environments (CVE) is one of the areas that have been in the spotlight. This has happened mainly because of two basically reasons: i) its possibility to use all the features of Virtual Reality, such as: navigation, interaction and involvement and allow the users to experience, through the computer and unconventional peripherals, real-life situations, and ii) the possibility for the users to share the same three-dimensional environment to exchange information, in order to enable a great variety of collaborative applications.

However, the conception of CVE is no trivial task, once it presents complex aspects in its construction. Therefore, these environments must satisfy a variety of characteristics simultaneously: fast reply to new system requirements; easy maintenance; interaction support in real time; high fidelity when it comes to the representation of the user; high rate of pictures per second; re-usage; portability, etc.

A great part of the problems related to the CVE has to do with communication support, which should allow a quick update of the interactions performed among the users. To minimize these difficulties, this work's main goal was the proposal of an intermediate layer of software, between the application and the communication layers, capable to, flexibly, support alterations and evolutions both in the application and in the infrastructure of communication, enabling the developer to direct his/her focus toward implementations of higher level processes, not worrying about changes in the communication layer.

As basic premises, this intermediate layer of software must enable easy update and creation of modules, and support a variety of applications.

Facing the relevance of the problem, this work initially characterized the necessary requirements to the conception of a CVE. In section 2, the requirements of a Collaborative Virtual Environment are presented, a model based on packages and exchange of messages aimed at the generalization of communication support between the collaborative virtual applications. In section 3, communication architecture for a CVE is presented. In section 4, the oriented modeling of the object of the architecture is presented. In section 5, the article presents its final considerations.

2 Collaborative Virtual Environments

Some of the most used definitions during conferences about the subject are presented as follows [1]:

"A CVE is a virtual space or a group of spaces based in one computer and distributed away. In this space, people are able to meet and interact with other people, with agents or virtual objects, in order to achieve a common goal. CVEs may vary in their representational richness, from 3D, 2.5D and 2D graphic environments to text-based environments. The use of a CVE is not lost in desktop devices, it may be well employed in mobile or wearable devices, public booths, etc."

This definition clearly sates that, although CVEs are normally associated with 3D graphic environments that are not always the case.

2.1 Characteristics of a CVE

The main characteristics of a CVE [1]:

- Shared Context consists in the users being able to share the environment, objects and information.
- Conscience of Other Individuals is the "understanding of activities of others which provide a context for their own activity" in a synchronic or asynchrony way.
- Negotiation and Communication respectively allow the delegation of tasks to the users of a common activity, and the exchange of information on the development of the tasks among them.
- Flexible and different points of view for more adequate decisions.

2.2 Requirements for a CVE Software

Simultaneous multi-user access is a necessary condition, but it is not enough to turn a virtual environment into a CVE. Several considerations on some aspects of hardware and software of the CVE are presented as follows [2] [1] [3]:

- Good Bandwidth;
- The Minimum Possible Latency;
- Centered Model of Communication presenting bottleneck, or, if distributed, presenting one extra layer increasing latency;
- Standard Internet Communication Protocol, TCP (Transmission Control Protocol) e UDP (User Datagram Protocol);
- Model of Storage that can be centered, problems with scalability, or distributed, where the virtual environment can be replicated or partitioned;
- Interaction may happen between user-world or user-user, and a good balance between interactivity and realism must be established;
- Good reliability of the critical data such as text messages. Some data are not so critical, as packages of voice or vectors of movement;
- Animations/Simulations must be based on frames so that there won't be loss of synchronism between machines of different computational capacity;
- Object sharing policy.

3 Architecture

3.1 Reasons and Goals

Multi-user and Distributed Collaborative Virtual Environments (CVE) are not completely new. The first work of great importance about them was the SIMNET in 1983, that ended up with the specification of the DIS (Distributed Interactive Simulation) architecture, and, later, of the HLA (High Level Architecture) [2] [3]. The latter is more flexible than the DIS, but both are directed toward military simulations.

Support architectures for distributed component oriented services, such as CORBA (Common Object Request Broker Architecture), have been taken into account when the implementation of distributed virtual environments took place [4]. However, the execution latency from a service request plus the latency caused by the remaining portion of the system could hinder the interactivity and the feeling of presence.

Commercially, there are several possibilities, as, for example, WorldtoWorld [4]. However, the majority of the tools, besides presenting a high cost, present problems with latency and scaling [4] [5].

Finally, if the Internet is taken for granted as a fitting means for user interaction in a shared virtual environment, it can become a mistake, once it's already part of many people's everyday routine, despite of some disadvantages presented (especially when it comes to the quality of service). Through the Internet, the research has been concentrated in VRML language towards the construction of tridimensional environments. This can be clearly verified because of the amount of existing environments and learning material that exist in the Web. Nevertheless, VRML language does not allow a great number of users to share a virtual environment. As a matter of fact, some proposals are in progress or in their conception phase. Some of them are listed as follows: LivingWorld [3], Virtual Reality Transfer Protocol (VRTP) [3], architecture DIS-Java-VRML [5], etc.

Based on the reasons presented above, this work proposes an architecture directed to the construction of collaborative virtual environments using the standard protocols of the Internet, once the majority of the papers developed in technical publications have focused on the conception of the communication inside the CVE [5] [6] [7]. This requires a great amount of time that could be spent in the development process of the application or in the setting of its requirements. When it comes to scalability and latency, they can improve if a hybrid architecture is used, optimizing the exchange of messages between client-server, as well as between server-server, different protocols and usage of dead reckoning tools, which aim to diminish the quantity of messages in the net, and deciding when a message should or should not be sent to other users.

3.2 Architecture Prototype

There are two main modules, the client module and the server module. Both will be detailed in the sub-sections that follows:

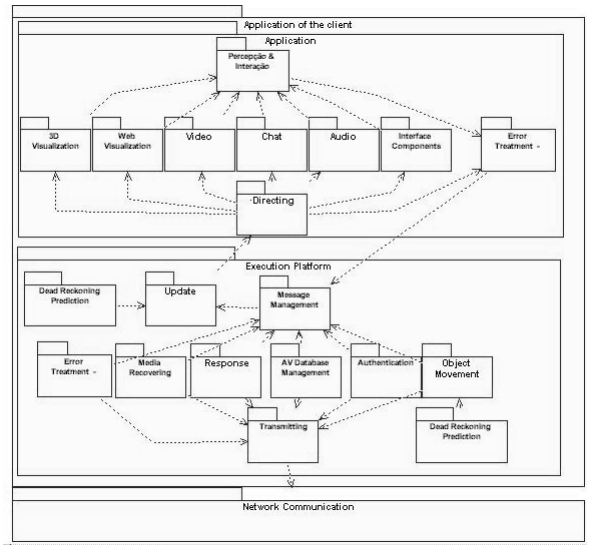

Figure 1: Diagram of Blocks of the Client Module.

3.2.1 Client Module

The client module (Figure 1) can be divided into two great parts, one is called the application, while the other one is called the execution platform. They will be detailed in sections 3.2.1.1 and 3.2.1.2.

3.2.1.1 Application

The submodules of the application of the client module are:

- 3D Visualization module and Web Visualization module are components that have already been created, such as Cortona (VRML player) and Internet Explorer (Web browser);
- Audio and Video module, in order to perform the communication among users or reproduction of media;
- Chat module enables text-based communication between users. It is the CVEs most used form of communication;
- Interface Components module: this module represents any object that can suffer interaction by the user, etc;
- Perception of Interaction module: all kinds of user interactions that are not treated by the modules to which they are interacting with, must be treated by this module. For example, voice recognition;
- Error Treatment module deals with errors in the manipulation of the application, and must be implemented by the developer of the application;
- Directing Module is in charge of receiving the messages deriving from the communication layer and updating the changes occurred in some components of the application.

3.2.1.2 Execution Platform

The sub modules of the execution platform are:

- Message Management Module, that is responsible for dealing with all the requests from the user and answers from the server, and decides to which module the message will be directed;
- Module of Update is responsible for directing the local and the remote updates (Message Manager) made by other users to the Directing Module of the application sub module;
- Media Recovering Module generates a message requesting the reproduction of a media that is stored in the server;
- Module of Object Movement is responsible for creating object movement updating messages in the VE;
- Module of Dead Reckoning is in charge of analyzing and discarding the object movement messages (in case the change in position is minimum), so that it could diminish the traffic in the net;
- Module of Dead Reckoning Prediction allows the movements of the objects to be continuous even when the messages are not sent through the net. It uses the two last sent messages in order to predict the next position of the object. When the module receives a position that is equal to one that it has already stored, or a pre-determined period of time finishes, then calculating new updates will not be necessary anymore;
- Database Management Module is in charge of the creation of the messages that request the recovery or storage of information in the database;
- Response Module is responsible for creating messages that enable other users to visualize the user's interactions, such as insertion and removal of objects. Moreover, it is responsible for transferring all the VE information when a new user enters;
- Authentication Module is responsible for creating messages that allow the user to get permission from the server in order to take part of a group, or environment, or to request his removal from the system;
- Receiving Module is responsible for receiving all the deriving messages from the communication net;
- Transmitting Module: this module is responsible for transmitting all the messages created by the aforementioned modules.

3.2.2 Server Module

The main purpose of the server is to be simple, and its main objective is to reply the messages in the most efficient way.

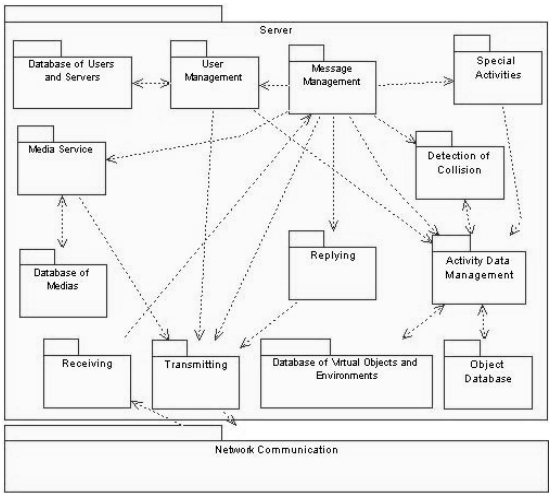

Figure 2: Diagram of Blocks of the Server Module.

In order to perform that, the server has the support of the following modules and database (Figure 2):

- Receiving Module is responsible for receiving all the deriving messages from the net, i.e., from the client machines;
- Message Management Module is responsible for identifying the type of message sent by the client, and then send it to the proper module;
- User Management Module receives the user's login and logout messages in the system;
- Media Service Module receives the messages for media reproduction requests, such as audio and video;
- Activity Data Management Module is responsible for receiving messages of modifications in the VE performed by the client, such as insertion and removal of objects. Moreover, it is responsible for keeping consistent instances of the virtual environment stored and to recover them upon request;
- Detection of Collision Module is responsible for receiving object movement information in the VE, identifying a possible collision, and sponsor status change in the involved objects;
- Replying Module defines to which clients the messages will be sent. The messages may be sent to one client, to all of them or to the group itself;
- Transmitting Module: this module is responsible for transmitting all the messages created by the previously described modules;
- Module of Special Activities is an optional module, and is useful if some programs cannot be executed by the means of an applet. The user then requests to the server the execution of this application or some other task. However, the main idea is not to implement any extra activity than the manipulation of messages by the server;
- Database of Users and Servers: this database stores information on the users and their roles, such as passwords, name, function, etc. It also stores information on the active servers who present similar contents, in order to redirect the user in case the server has reached its limit number of users. Every time There is an inclusion of a new user, the information is repassed to the other active servers so that their databases are updated;
- Database of Medias is responsible for storing information on the medias that can be reproduced;
- Database of Virtual Objects and Environments is responsible not only for storing the semi constructed object models, but also for storing information about the changes in the virtual environment caused by interactions of the users. Moreover, it must keep a consistent copy of the shared virtual environment.

4 Modeling of the CVEs Architecture

Based on the architecture shown in section 3, this section will present the object-oriented modeling for the architecture. The purpose of its presentation through the Unified Modeling Language (UML) is to enable the validation of the architecture, as well as make it easier for its understanding.

The diagrams of the UML selected to present the architecture are [8] [9]: the Diagram of Packages, in order to represent the general vision of the system's architecture; the Diagram of Classes, as a way to represent the connections between the manipulable elements of the architecture; the Diagram of Sequence, to represent the interactions (message exchange) among objects; and Diagram of Execution, to represent the environment (operational system, communication protocols and configuration of the machine).

4.1 Diagram of components

Figure 3 presents the diagram of packages [8] [9] - General Vision, as a way to show the integration among the two views of the architecture:

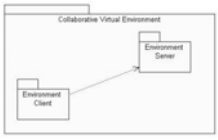

Figure 3: Diagram of packages - General View of the CVE

4.2 Diagrams of Usage

Inside the Collaborative Virtual Environment context, the requirements to be implemented are described as taken

from the diagram of each case of usage [8] [9], first illustrated in table 1. Its graphical representation are presented in Figure 4 and Figure 6.

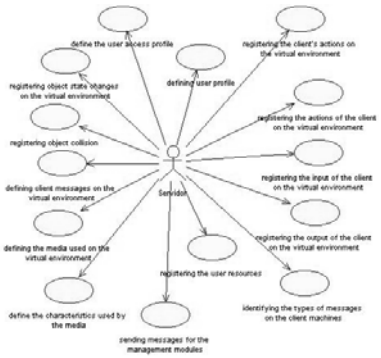

Figure 4: Diagram of Case of Usage for the Server Agent.

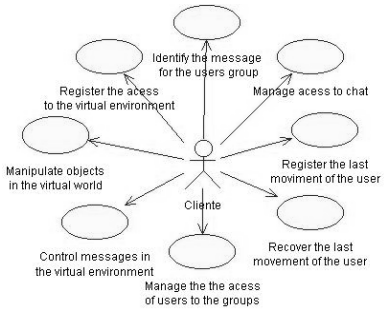

Figure 5: Diagram of Case of Usage for the Client Agent.

4.3 Diagram of Classes

As a way to represent the data components and its interactions in the Collaborative Virtual Environment a diagram of classes is presented here [8] [9]. Its details are illustrated in Figure 7.

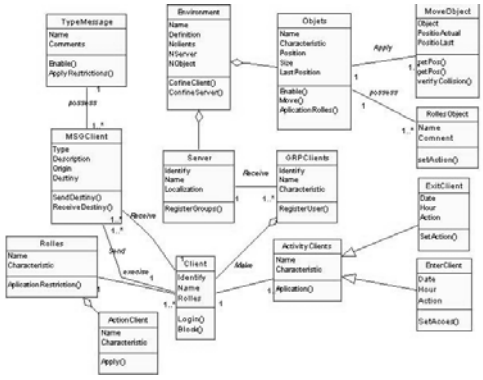

Figure 6: Diagram of Classes of the Collaborative VE.

4.4 Diagram of Sequence

The diagram of sequence shows the exchange of messages between objects in one determined case of usage. Two cases of usage are presented in Figures 8 and 9. They demonstrate the interaction between the main elements in the context of the CVE [8] [9].

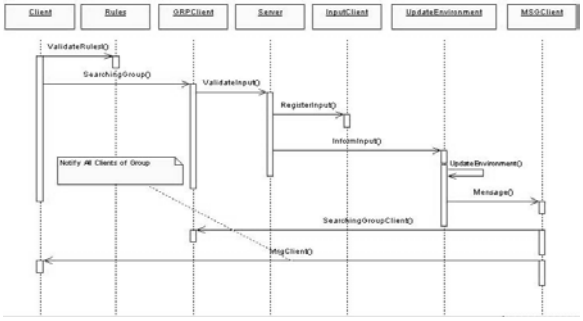

Figure 7: Diagram of Sequence for Client Authentication.

The Authentication of the Client in the Collaborative Virtual Environment represents the moment when each client identifies himself. It is important to point out that each client is associated with a group of clients, and this group is linked to a server (Figure 8).

Another action of the user is the possibility to insert objects. The synchronism among the users of the shared environment must be kept (Figure 9).

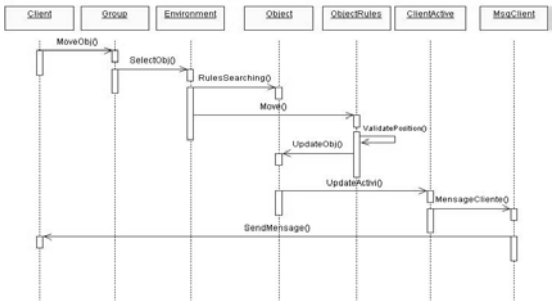

Figure 8: Diagram of Sequence in order to insert an Object.

4.5 Diagram of Execution

Figure 9 presents the diagram of execution, which shows the necessary resources to support a CVE.

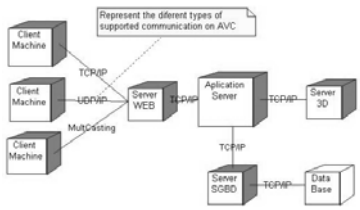

Figure 9: Diagram of Execution of the Collaborative VE

5 Final Considerations

The architecture proposed was conceived to fulfill the following properties of software engineering:

- Generality: the architecture enables the use of several applications with different requirements. For instance, there are applications that use text as a form of communication between users, while others use audio. There are applications that use unconventional

peripherals, such as gloves and stereoscopic eyeglasses, and others that use only the keyboard and a mouse. All the applications can be contemplated by the architecture;

- Modularity and Flexibility: the architecture is made up of modules. They enable a fast inclusion of new characteristics through the implementation of new modules. Moreover, they allow an easy adaptation of the existing modules to support some specific requirement of other application.
- Portability: the architecture enables the development of the applications in any platform, once the architecture will be conceived with standard and portable technologies.

In terms of functionality, the architecture shows the following characteristics:

- Ease of Applications Construction in Collaborative VR: one of the architecture goals is to allow new collaborative virtual applications to be developed in less arduous way, since the complexity of the communication will be distraught by the architecture.
- Easy Upgrade from Single user VR Application to Collaborative VR Applications: another goal of the architecture is to facilitate the upgrade of Single user VR applications, created in portable language, to collaborative VR applications.
- Low Demand for Bandwidth: since the virtual environments will be shared through the Web, the balance between realism and interaction must be sought. Thus, the architecture provides very simple messages, which are optimized in size, tending to increase interactivity and to diminish realism. The usage of different protocols to treat different data also helps in a lower use of the bandwidth.
- Low Latency and Some Levels of Reliability: it is mainly related to the realism of the VE. The lesser the need for bandwidth the lesser the latency and its variation. Moreover, the usage of different protocols for different types of messages is appreciated. For example, placement messages (non critical) and text messages of communication (critical). The usage of a reliable protocol is recommended for critical messages, therefore the loss of some part of the information can jeopardize its total comprehension. Other important topic is the usage of filtering mechanisms of information, such as limited broadcast (not all messages are sent through the net), dead reckoning (prediction of movement by the remote user), etc.
- Standard Communication Protocols and Models: the usage of well-defined and standardized application programming interfaces makes the architecture sufficiently adaptable to changes in the communication protocols. Moreover, the architecture must be ready to work with the protocols of transport of the Internet (TCP, UDP), in the unicast, multicast and limited broadcast types.
- Centered Model of Storage: the architecture provides a model of storage classified as centered, which is easier to manage. It presents some scaling problems when the number of users ranges from average to high (DIEHL, 2001). There were some efforts tending to minimize this problem with the creation of groups to avoid server overload, thus there are several instances of servers with a maximum number of users. However, a completely distributed approach also presents inconsistency among the users, besides presenting a complex management, and this can be a problem because of the requirements of the application.
- Manipulation of Shared Objects: the architecture provides two mechanisms of shared objects. Either the creator is the only one responsible for the manipulation of the object, or any user can manipulate the object in relation to its time request.

6 References.

[1] Churchill, E. F., Snowdon, D. N., Munro, A. J. Collaborative Virtual Environments: Digital Places and Spaces for Interaction. Springer. 2001.

[2] Singhal, S. Zyda, M. Network Virtual Environment – Design and Implementation. Addison Wesley. 1999.

[3] Diehl, Stephan. Distributed Virtual Worlds: Foundations and Implementation Techniques Using VRML, Java and Corba. Springer. 2001.

[4] Kubo, M. M. – "Suporte de Comunicação para Sistemas Colaborativos Interoperáveis de Realidade Virtual", Masters project, UFSCar, São Carlos - SP, 2000.

[5] Reis, D. S. dos; et. al. "DIS-Java3D para Ambientes Interativos Distribuídos em Rede". Proceedings of the 5th SBC Symposium on Virtual Reality (2002), Fortaleza-CE, 103-114.

[6] Perucia, A. Pinho, M. Ferramenta de suporte à construção de ambientes virtuais colaborativos. Proceedings of the 5o SBC Symposium on Virtual Reality (2002), 115-125. Fortaleza – CE. 2002.

[7] Leite, A. J. M. Jr., et al. Um Ambiente Virtual Compartilhado Voltado para o Entretenimento. Proceedings of the 5o SBC Symposium on Virtual Reality (2002), Fortaleza-CE, 138-149.

[8] LARMAN, Craig. Utilizando UML e Padrões – Uma Introdução à análise e ao projeto orientado a objetos, Porto Alegre: Bookman 2000.

[9] BOOCH, Grandy; Rumbaugh, James; Jacobson, Ivar. UML: Guia do Usuário – Rio de Janeiro: Campus 2000.