Proceedings of the 5th WSEAS Int. Conf. on Signal Processing, Computational Geometry & Artificial Vision, Malta, September 15-17, 2005 (pp197-201)

Towards a Model and a Framework to Build Web-based 3D Collaborative Virtual Environments Populated by Interactive Entities

Rolando Menchaca Méndez[1], Leandro Balladares Ocaña[2], Ruben Peredo Valderrama[2], Chadwick Carreto Arellano[2]
[1]University of California Santa Cruz, School of Engineering, 1156 High Street,
95064 Santa Cruz, CA, USA

[2]National Polytechnic Institute, Computer Science Research Center,
07738, Mexico City, México

Abstract: - In this paper a model and a Java-based framework to aid the construction of 3D Web-browsing collaborative virtual environments (3D-CVE) populated by interactive entities are described. Our proposal emphasizes on the collaboration aspects among the entities that populate the virtual world and the services that they offer each other in order to carry out collaboration, rather than the modeling and aesthetical aspects of the worlds. We propose a model for the conceptualization of the virtual world under the concept of social groups, a graph-based high level notation to specify the interactions among the entities, and a Java based software framework that gives support to the model and the interaction graph in order to facilitate the implementation of the CVE. By means of a directed graph, the model describes interaction between the entities that populate the virtual world. The nodes and edges of the graph can be mapped to entities (classes and interfaces) of the proposed architecture, reducing the time and effort needed to develop this type of applications. The architecture allows for easy distribution management of processes between clients and a server, or otherwise its centralization within a central server. We describe the current implementation and an example application.

Key-Words: - Collaborative Virtual Environments (CVE), 3D-CVEs, Software Engineering, Design Patterns, Web-based 3D-CVE

1 Introduction

Collaborative Virtual Environments (CVEs) are defined as: "…a computer-based, distributed, virtual space or set of places. In such places, people can meet and interact with others, with agents or virtual objects. CVEs might vary in their representation richness from 3D graphical spaces, 2.5D and 2D environments, to text-based environments…" [1]. Those CVEs that use 3D graphical spaces as a representation mean are knew as 3D-CVEs. In the rest of the paper we will use CVE to refer to 3D-CVE. For this work, the proposed CVEs are composed by three elements: individuals, artifacts and decorations. Individuals can be users' avatars or autonomous entities that interact within the CVE through a service based concept. The individuals define the actions that users or agents are able to perform. Artifacts are elements that individuals can interact with, their services are implemented by software components and they aren't embodied by any one user but, unlike agents, these don't realize some task in autonomous way unless some individual request them through one of the services they offer. These artifacts enable collaboration tools such as shared content editors, blackboards, etc. They are also the access point to other kinds of services such as e-commerce transactions. Decorations are static objects that are visible within the virtual world but do not have collaboration interface. As in a real world, individuals sharing common characteristics can be classified into social groups. In this case, the characteristics that define the property of a group are the individuals' abilities and the way in which they interact with other components of the CVE (individuals or artifacts). Thus, as it is shown in Fig. 1a, the worlds are populated by sets of individuals pertaining to different social groups and by sets of artifacts that provide some type of service.

The set of all existing social groups within a CVE is denoted as *S*, and the set of all types of artifacts as *A*. Each group partially defines the way its elements interact with other group elements. In order to totally define the interactions that can be carried out within the CVE, a directed graph $G=(V,E)$ where $V=S\cup A$ and $E\subseteq S\cup A\times S$ is proposed, and where the edges represent a service relationship labeled with a pair $Event_i$ / $Interface_{'a'?SG\ TG}$, where: $Event_i$ is the event that will trigger the collaboration between entities from *TG(Target Group)* with entities from *SG (Source Group)*; if *a* is used, then *SG* is an artifacts group; $Event_i$ = type of event = *{Proximity (P), Inside Of (IO), Right Click Over (RCO), Left Click Over*

(LCO), etc.}. The relations defined in the graph are translated into graphical interfaces that will appear in the browsers of the users when their avatars (or the avatars of some other individual) produce a collaboration event. As it is shown in Fig. 1b, if an edge connects the social group G1 to group G2, it indicates that, when P occurs (when the G2 entity be "near"- 'near' is defined in terms of the auras of the geometrical representations of the entities. An aura is an imaginary sphere covering the avatar. Two entities are near each other when their auras intersect them in the space - to the G1 entity) an individual pertaining to group G1 will offer a service interface (I_{12}) to an individual pertaining to group G2. When the individuals 'move away' (the auras don't intersect them) from each other, it is assumed that the collaboration has finished, and the graphical interface that enables the collaboration will disappear from the browsers of G2 entity. So, the process of executing an action that is defined in the collaboration interface of another individual is denominated as a collaboration action.

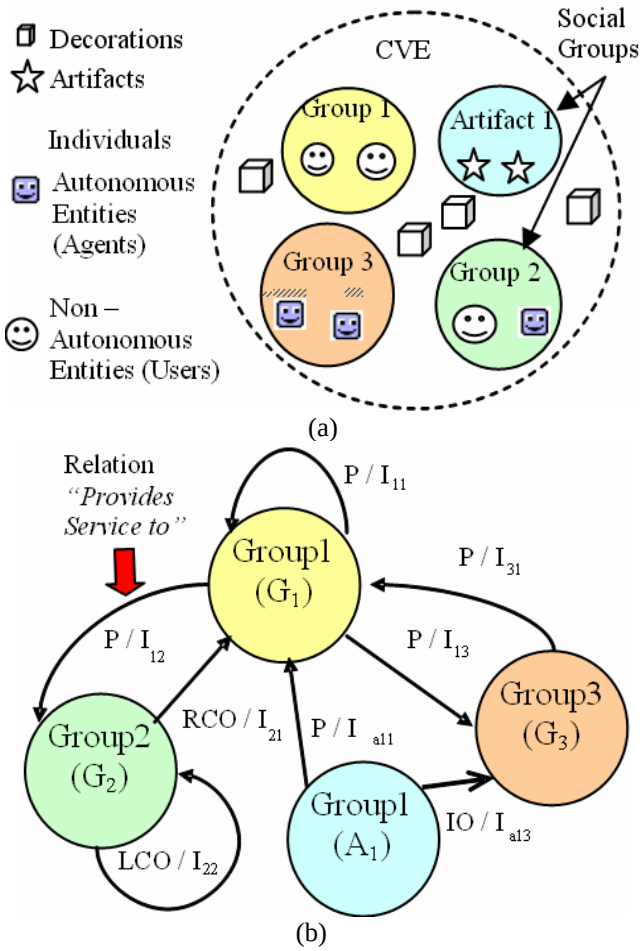

Fig. 1. a) CVE conceptual model; b) collaboration graph: a directed graph defining the collaboration relationships in a CVE.

Each one of the edges of G defines a particular pair *Event*$_i$ / *Interface*$_{'a'?SG\ TG}$. This allows an individual to collaborate in a particular way depending on with whom he is collaborating. As a result of the use of the services exposed by an artifact, an individual can modify his state, the state of some other individual, the state of the artifact, some of the data bases related to the individual or some other individual, or any other aspect modeled within the CVE. Under this scheme, it is possible to divide the CVE into different regions (modeled like artifacts) where the individuals acquire new abilities when entering (they can collaborate with the region).

2 The Framework

The CVEs are mainly made up of three classes: RemoteSoul, CollaborativeWorld and CollaborativeApplet. The RemoteSoul class represents the state and behavior of the individuals and implements the social groups defined in the collaboration graph. A CollaborativeWorld class serves as a meeting point for the participants and, as explained further, as a container of souls and references or just as a container of references. The CollaborativeApplet class represents the user's environment. It is the class where rendering is carried out and depending on the system architecture (centralized or distributed) the processes associated with the soul of the individual that represents the user or agents are made available (see Fig.2).

2.1 The Souls

Artifacts and individuals are made up of three elements: geometry, state and behavior. The geometry is the way in which individuals or artifacts looks like within the CVE. The state, as well as part of the behavior, is encapsulated within a remote object that implements the soul of the object. Every social group defined in the collaboration graph must be mapped into an interface that specializes the RemoteSoul interface, or in other words, to a specific kind of soul. Figure 2 shows three classes of souls for non-autonomous entities (SubjectType1, SubjectType2 and SubjectType3) and three classes of souls for autonomous entities (AgentType1, AgentType2 and AgentType3), that respectively implement interfaces RemoteType1, RemoteType2, RemoteType3, RemoteType4, RemoteType5 and RemoteTypeN. Some of the subject's abilities are implemented within its soul. These abilities are implemented by a set of remote methods. For the case of the non-autonomous entities the soul exports to its users these abilities by means of graphical interfaces

that are displayed in the user environment (CollaborativeApplet). For autonomous entities these abilities are used during the implementation of their behaviors. In general, a soul (an element of a particular social group) implements three different kinds of interfaces: 1) those related to the abilities inherent to the subject (HS), meaning all the capacities that a subject has during its lifetime within the world and that do not depend on the interaction with other individuals or artifacts; 2) those related to the collaboration actions (HC), meaning that each one of these interfaces, is associated with one edge leaving from the social group in the collaboration graph; those related to the collaboration actions that other individuals provide but that modify its state (SI), associated with an incoming edge to the social group in the collaboration graph.

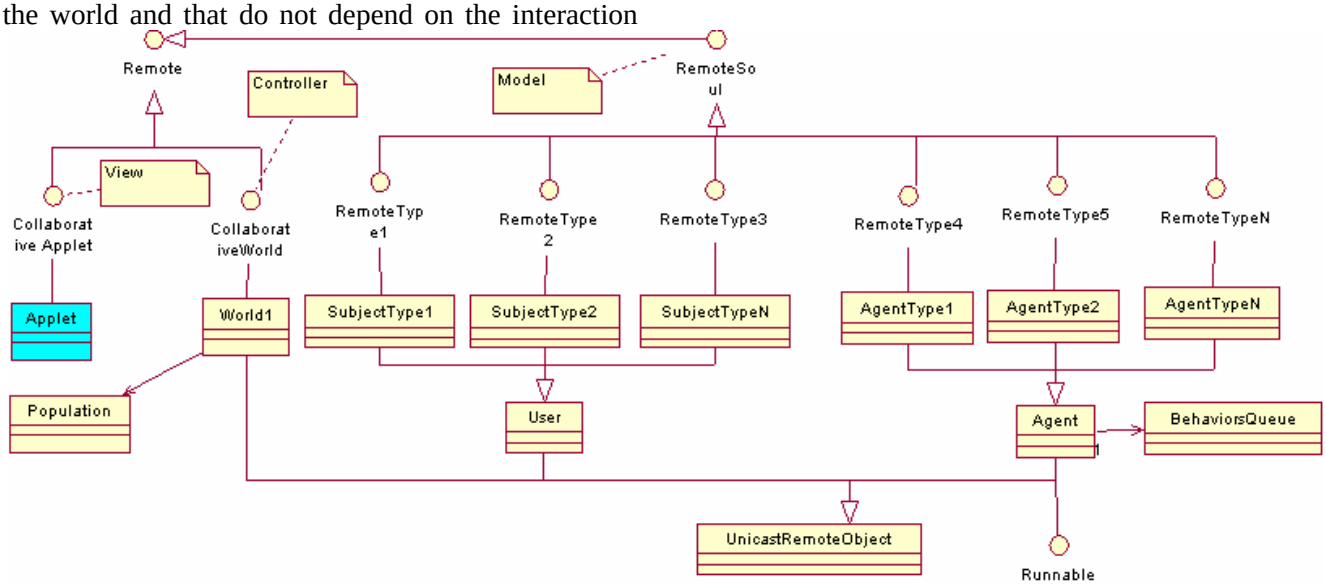

Fig. 2. Main interfaces of the framework.

The set of actions that an individual is able to perform during its stay in the CVE is made up of all the methods implemented in its soul, plus all the collaboration actions that other individuals or artifacts can do in its name. All these actions, for the case or artifacts and non-autonomous entities, will be performed as an answer to an explicit request of the user by means of some of the controls displayed in the graphical interfaces; for agents these actions are performed when a behavior is executed.

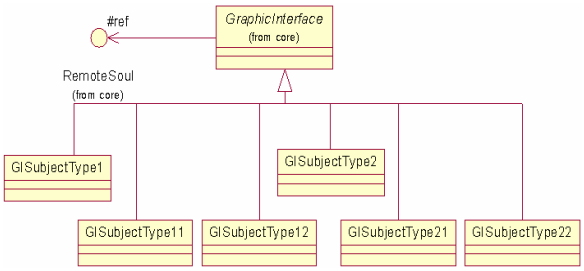

Fig. 3. Diagram of classes of the social groups G1 and G2.

All the graphic interfaces must specialize the abstract class *GraphicInterface*, in order to isolate the subjects and the collaborative applets from the details of the interfaces defined by other social groups. Fig. 3 shows the summarized class diagram of the graphic interfaces for two hypothetical social groups.

When a collaboration event takes place, souls construct the suitable graphical interface (that corresponds to the type of subject with whom it is collaborating) and send it to the target *CollaborativeApplet*. To dynamically identify the social group of a subject or the type of an artifact, we use the reflection support within the Java language.

The framework enables the use of different actualization schemes for the entities of the world (Identifying actualization schemes is very valuable because there is not a best actualization protocol for every situation [2], [3], [4]). The proposed architecture allows the implementation of three different schemes to distribute the data and behavior in a CVE: centralized, distributed and hybrid. In the distributed scheme the amount of communication resources grows linearly in respect to the number of users in contrast with the quadratic grows of the centralized schema. Actualizations are made directly from client to client, whereas in the centralized schema the invocations are directed to the server and then to the clients. The feedback is improved because the remote object that implements the user's avatar is in the same address space (in the Web browser). Thanks to the encapsulation of the code involved in the update protocols, the implementation of a hybrid scheme is straightforward. In the prototypes, we

include a check box where the user can locate the soul of his avatar. In the future, this decision can be made automatically based on objective parameters such as hardware and network characteristics, number of users, world behavior, etc

2.2 Collaborative World and Collaborative Applet

The main task of the CollaborativeWorld object is to keep a record of the participants of the CVE. These objects generally reside on the same machine as the Web server that hosts the world. Users access the CVE by means of a Web browser. The browser accesses a HTTP server where it obtains the CollaborativeApplet (CA) that is in charge of executing all the necessary code to participate in the CVE.

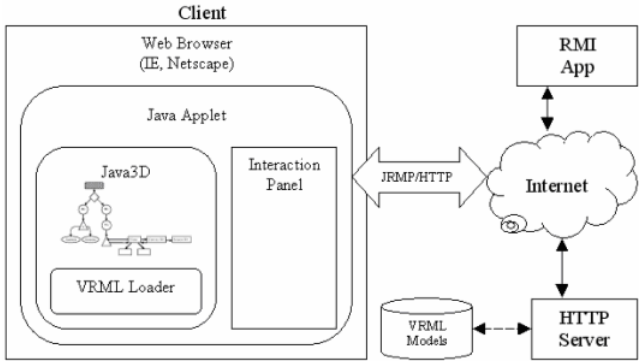

Fig. 4. The CollaborativeApplet serves as bridge between the technologies in charge of rendering the CVE and the objects that keep their state and implement their behavior.

The Fig. 4 shows an example of a CollaborativeApplet. As was mentioned, the CollaborativeApplet is in charge of contacting the CollaborativeWorld to get the code necessary to interact within the MVC, meaning references to the souls of all participants and a reference to the soul of the user's avatar (or a factory to construct the soul within the applet). Within the Applet, threads are constructed to obtain the state of the objects that implement the souls of subjects and artifacts. We can use different update schemes depending on the nature of the application: 1) Pooling: a thread is associated with the reference of each active element of the CVE. The thread performs a cycle where it invokes a remote object method to obtain his state. With this information it updates the appearance of the element in the VW. 2) Update: when the state of a soul is modified, it makes calls to its references (callbacks) so that clients update the appearance of this element in the VW. 3) Both Sides Processing: the code that is in the browser implements part of the state and behavior of the souls. Local objects make calculations associated with the behavior of the subjects but they maintain communication between them for synchronization

3 Conclusions and Future Work

To test the framework, we developed a very simple CVE of a virtual shop in which only will have two types of users: sellers and customers (see Fig. 5). In this first implementation the shop is empty and only sellers and consumers habitat it. Consumers and seller can communicate through textual messages, like in a chat (in a more complex application could be used the Java Voice XML API in order to support voice communication).

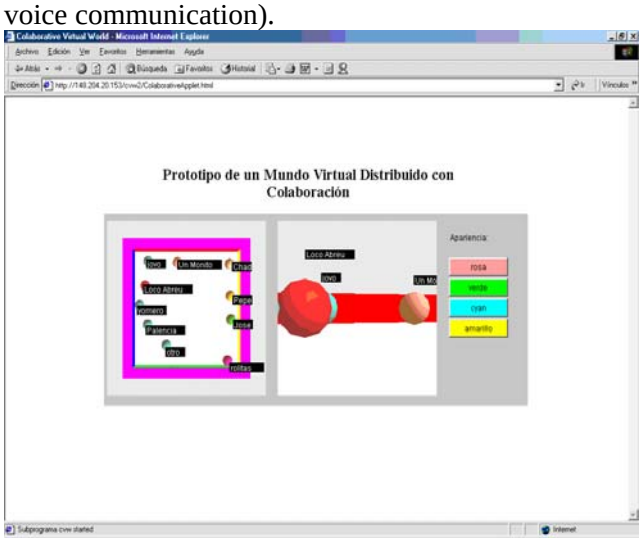

Fig. 5. Client view of the CVE.

From the developing process point of view, the goal of the collaboration model and framework is not only to develop CVE faster, but also the resulting CVE have similar structures. They are easier to maintain and eventually to integrate. The current implementation support non-autonomous entities, we have made the design for the autonomous entities and we are working in its implementation. As a future work, we must work in to extend this to support validation of new abilities for entities at run-time; also we must work in security, quality of service (QoS) and persistence aspects applied to CVE. We are working too in to improve the model and the framework in order we can develop more interesting applications, such as virtual laboratories, which were one of the reasons we had started this work.

References:

[1] Elizabeth, F., David, N., Alan, J. (Editors), Collaborative Virtual Environments: Digital Places and Spaces for Interaction, Springer-Verlang, 2001.

[2] Atul, P., Hyong, S., Jang, H., Data Management Issues and Trade-Offs in CSCW Systems. IEEE Transactions on Knowledge and Data Engineering, Vol. 11 No. 1, January/February, 1999.
[3] Hans-Peter, D., Garcia-Luna-Aceves, J., Group Coordination Support for Synchronous Internet Collaboration. IEEE Internet Computing, March-April, 1999.
[4] Vidot, N., et-al.: Copies Convergence in a Distributed Real-Time Collaborative Environment. Proceeding of the ACM 2000 Conference on Computer Supported Cooperative work. Philadelphia, Pennsylvania, United States, 2000.