Design of a Layered Component-Based System for Sharing Visualization Objects in Collaborative Environments

Mario CIAMPI

Institute for High Performance Computing and Networking, National Research Council of Italy
Via P. Castellino, 111–80131 Naples, Italy

Abstract: This paper presents a layered component-based system approach aiming at providing facilities for cooperating in Collaborative Visualization Environments by sharing the state of the visualization between users. The requirements addressed by the system concern scalability, decoupling, abstraction, and application-independence. The proposed component-based system is structured into two different layers: (i) Collaborative Middleware Layer, responsible for notifying users when events occur, and (ii) Collaborative User Layer, in charge of intercepting and executing user requests. The paper presents also a collaborative task flow diagram, which shows the sequence of procedures that must be completed to create and share visualization objects. The applicability of the system is illustrated by a healthcare case study, in which two users interact with each other as well as with visualization objects in a shared collaborative environment. The visualization objects are generated by applying two different reconstruction techniques to medical images coming from various modalities like Computed Tomography and Magnetic Resonance Imaging: Ray Casting and Multiplanar Reconstruction.

Keywords: Component-Based System, Collaborative Visualization Environment, Event Notification, Task Flow, Healthcare, Reconstruction Techniques.

1. INTRODUCTION

Collaborative Visualization Environments (CVEs) are communication systems in which multiple users get immersed into a virtual environment in order to work together by visualizing shared virtual objects. CVEs are increasingly being used for numerous application domains such as e-learning, collaborative design and engineering, multiplayer games, and telemedicine [1].

Information sharing is crucial for effective collaborative work [2][3]. In fact, participants in collaborative sessions can interact with (i) each other and (ii) data representations (visualization objects) by sharing their results. For this reason, collaborative systems aim at providing the maintenance of a consistent shared state across the collaborating organizations [4].

According to the Applegate’s place-time matrix [5], the participants to a collaborative session can be geographically co-located or distributed and can cooperate in a synchronous or an asynchronous way. Thus, the collaboration among users may vary over time, even if it is essential to respect some requirements. Among these, it is fundamental to share information in a consistency and responsive way. As a matter of fact, a key challenge in collaborative environments is to allow cooperating participants to simultaneously visualize the same scene by maximizing responsiveness and ensuring the best use of limited and dynamic resources that can bring significant delays [6]. These issues can be overcome by designing opportune software architectures that permit the users to share with each other what that each sees on his/her screen, according to the paradigm known as WYSIWIS (What You See Is What I See) [7].

Over the last few years, several approaches have been proposed for this purpose. CSpray [8] is a collaborative 3D visualization system, where users can change the type of the visualization generated by means of spray cans, which are fired into data volume workspace. SGI OpenGL Vizserver [9] is a solution developed by Silicon Graphics, Inc. designed to enable remote-visualization applications. It uses algorithms to compress and decompress images in the scene. Access Grid [10] is a project of the NCSA Alliance which aims at creating virtual spaces where persons can cooperate with each other. These virtual spaces offer a variety of features, such as shared desktops, presentation materials, and so on. Several approaches have been proposed also to ensure consistency in CVEs: the current state-of-the-art technique is based on dead reckoning mechanisms [11], even if other
methods are available [12]. Such systems tightly couple the collaboration layer to the rest of the visualization system.

This paper focuses on the question of how to share the state of the visualization among distributed collaborative systems, despite the technique used to reconstruct visualization objects [13]. To reach this goal, the paper proposes a component-based system approach that aims at notifying the participants to a collaborative session about the actions executed by the other users by means of a message passing paradigm. The proposed system, organized into two layers, takes into account that users can interact with different viewers on the basis of the reconstruction technique used. This aspect is relevant, since the users can continue to visualize the objects in the way they prefer.

This rest of this paper is organized as follows. Section 2 presents the proposed component-based system, by highlighting the requirements addressed and the software components. In Section 3, a collaborative task flow diagram is described. A healthcare case study is discussed in Section 4. Finally, Section 5 concludes the paper.

2. LAYERED COMPONENT-BASED SYSTEM DESIGN

This section presents the design requirements addressed and the software components that make up the proposed system.

2.1. Design Requirements

The approach adopted to support the implementation of CVEs consists in a component-based system that features the following properties:

- **Scalability:** collaborative environments are characterized by a strong dynamicity, due to the fact that users can join and leave collaborative sessions in any time.
- **Decoupling:** participants have to cooperate with each other in a decoupled fashion, rather than by interacting directly.
- **Abstraction:** the advantages of abstraction concern mainly the hiding of implementation details, software reuse, and simplification.
- **Application-independence:** the system has to be designed in order to suit any applications.

2.2. Software Components

Fig. 1 depicts the proposed component-based system, that is structured into two independent but interconnected layers:

- **Collaborative Middleware Layer:** this layer is the cornerstone of the system, since it aims at enabling applications located in the upper layer to communicate with each other by means of event notification.
- **Collaborative User Layer:** at this layer there are application-specific components, that intercept the user requests and propagate them to the virtual environment by means of the Collaborative Middleware Layer.

![Proposed Two-tier Component-based System](image)

Figure 1: Proposed Two-tier Component-based System

The software components of each layer are described in detail below.

The **Collaborative Middleware Layer** is composed of the following software components:

- **Notification Broker:** this is the key element of the system. It aims at dispatching the events received to all the interested consumers. The event notification is based on the topic-based publish/subscribe paradigm [14]. Thus, a consumer interested to receive a specific kind of events has to subscribe to the Notification Broker for a particular topic. In this way, this component decouples source systems and target systems.
- **Collaborative Session Manager:** this component is responsible for handling the collaborative sessions. In greater detail, the functionalities provided allow to create, join, and leave collaboration sessions and to require the control of the interaction.
- **Floor Control Manager:** this component manages the user accounts of the CVE, by enabling or denying the access to the requesting users.

The **Collaborative User Layer** consists of the following software components:
Design of a Layered Component-Based System for Sharing Visualization Objects...

- **Repository Manager**: this component aims at interacting with data repositories in order to make the data available for the visualization.

- **Viewer Selector**: the aim of this component is to load the opportune viewer on the basis of the type of the data required, the user preferences, etc.

- **Visualization Manager**: this component is in charge of visualizing virtual objects according to a specific technique. There can be more than one of such a component, in accordance to the type of the data to visualize.

- **Interceptor Manager**: all the user requests are intercepted by this component and sent to the virtual environment by means of the Communication Manager.

- **Communication Manager**: the aim of this component is to (i) propagate to the virtual environment all the user requests received by the Interceptor Manager, and (ii) interact with the Repository Manager and the Viewer Selector after having received events incoming from the underlying Notification Broker.

3. **COLLABORATIVE TASK FLOW**

The software components described in the previous section interact with each other in order to execute the tasks depicted in the diagram shown in Fig. 2.

![Collaborative Task Flow Diagram](image)

As shown in the figure, the whole Collaborative Task Flow Diagram can be grouped into two main subsystems: **Visualization SubSystem** and **Collaboration SubSystem**. The tasks located in the former subsystem are aimed at handling visualization issues, whereas the latter subsystem is focused on managing aspects related to collaborative sessions. Every task is described here:

- **Retrieve Data**: according to the user requests, this task makes available for the visualization the desired data by requiring the Repository Manager to interact with the opportune data repository by means of a specific communication protocol.

- **Matchmake**: this task has the aim of determining the appropriate viewer to load on the basis of the type of the data to visualize and the user requests. This task is accomplished by the Viewer Selector component.

- **Load Visualization**: the goal of this task is to require the Visualization Manager to load a specific viewer and to solicit the visualization of the selected data.

- **Visualize Data**: after that a viewer is loaded, the visualization of the requested data can be performed. Also this task is performed by the Visualization Manager component.

- **Interact**: this task acts as a bridge between the two subsystems. Its main goals are to i) allow a user to interact with the visualized data (e.g. moving and rotating virtual objects, but also joining existing collaborative sessions), and ii) require the notification of interaction events to the environment. It is also responsible for requiring the execution of tasks on the basis of the requests coming from the other users. This task is performed by orchestrating the Interaction Manager, Visualization Manager, and Communication Manager components.

- **Create/Join Collaborative Session**: these tasks enable users, respectively, to create and join a collaborative session. Their execution is managed by the Collaborative Session Manager component.

- **Leave Collaborative Session**: this task enables users to leave a collaborative session. It is performed by the Collaborative Session Manager component.

- **Require Control**: this task is responsible for managing the floor control, that is, it allows or not a user to take the control of the interaction according to specific policies. It is handled by the Floor Control Manager component.

- **Notify/Receive Events**: these tasks are in charge of maintaining the distributed visualizations in a consistent state by sending and receiving...
event notifications through the Notification Broker component.

4. A HEALTHCARE CASE STUDY

This section illustrates a case study about the applicability and usage of the proposed system.

In the case study considered, two users connected to two different nodes interact with each other by sharing a consistent virtual environment. In this environment, the users visualize the same virtual objects, obtained by reconstructing medical images coming from different scanning techniques, such as Computed Tomography (CT) and Magnetic Resonance Imaging (MRI). It is assumed that the data are archived in shared repositories, accessible by all the participants.

Here, the whole scenario is described step-by-step.

1. The first user interacts with the system in order to create a collaborative session by means of the task \textit{Create Collaborative Session}.

2. The second user joins the collaborative session by performing the task \textit{Join Collaborative Session}.

3. The first user requests the system to retrieve medical images archived in a specific data repository. The task \textit{Retrieve Data} is performed.

4. The first user asks the system to reconstruct and visualize the medical images by using a particular technique, available on the local node. The tasks \textit{Matchmake}, \textit{Load Visualization}, and \textit{Visualize Data} are executed. Now, the first user has the control of the interaction, as he/she is the creator of the collaborative session.

5. The first user interacts with the virtual objects that he/she sees on his/her screen by requiring the execution of the task \textit{Interact}. A number of messages are sent to the software components of the \textit{Collaborative Middleware Layer} by performing the task \textit{Notify Events}.

6. The second user is notified of all events that occur on the first user's node through the task \textit{Receive Events} and all the actions executed on the second node are performed also on the first one.

7. The second user requires the control of the interaction by executing the task \textit{Require Control}.

8. The second user interacts with the virtual objects that he/she sees on his/her screen by requiring the execution of the task \textit{Interact}. A number of messages are sent to the software components of the \textit{Collaborative Middleware Layer} by performing the task \textit{Notify Events}.

9. The first user is notified of all events that occur on the second user's node through the task \textit{Receive Events} and all the actions executed on the second node are performed also on the first one.

10. The second user leaves the collaborative session. The task \textit{Leave Collaborative Session} is performed.

11. The first user leaves the collaborative session by requiring the execution of the task \textit{Leave Collaborative Session}.

The interesting aspect is that the participants to the collaborative session can visualize the reconstructed images by using the technique they prefer. The reconstruction techniques considered in this case study are Ray Casting and Multiplanar Reconstruction (MPR). They are extensively used in the field of medicine for the purpose of diagnosing diseases.

Ray Casting is a technique used to visualize a 2D projection of a 3D data set, whereas MPR is a method aiming at reconstructing images in order to be displayed in any planes (such as axial, sagittal, and coronal).

All the actions are notified to the environment by sending the pair \textit{<action, parameter>}. The possible actions are the following:

- \textit{<rotate, clockwise/anticlockwise>:} this action rotates an object.
- \textit{<move, up/down/left/right>:} with this action it is possible to move the virtual objects in the space.
- \textit{<zoom, in/out>:} this action performs zoom operations.
- \textit{<apply CLUT, clut>:} the aim of this action is to apply a Color Look-Up Table (CLUT) to the images.
- \textit{<adjust WL/WW, wl/ww>:} this action makes it possible to adjust the window level (WL) and window width (WW) for a data representation by changing a transfer function.

Fig. 3 and Fig. 4 show the reconstructions of two CT datasets obtained by using, respectively, Ray Casting and MPR techniques.
5. CONCLUSIONS

This paper has presented a layered component-based system for Collaborative Visualization Environments, which are distributed systems where users can interact with each other by visualizing virtual objects. The proposed system provides basic components that enable remote users to share the state of their visualization through the event notification paradigm. Specifically, the system is partitioned into two tiers, namely Collaborative Middleware Layer, responsible for dispatching messages, and Collaborative User Layer, in charge of intercepting and executing user requests.

A Collaborative Task Flow Diagram showing the tasks performed during the interaction of the software components of the system has been illustrated.

The applicability of the proposed systems has been examined in a healthcare case study, where two users cooperate with each other by visualizing shared medical images reconstructed using two reconstruction techniques: Ray Casting and Multiplanar Reconstruction.

Future work will regard the evaluation of performance measures (such as time consumption). In addition, the application of the proposed system to a more exhaustive case study, including image processing techniques such as image fusion [15] and image segmentation [16], has been planned.

REFERENCES


