IX Simpósio Brasileiro de Sistemas Multimídia e Web - WebMidia 2003, p.353-367. Salvador, BA, Brasil. 2003.

CSVTool – A Tool for Video-Based Collaboration

Cesar T. Pozzer[1], Alberto B. Raposo[1], Ismael H. F. Santos[1, 2], João L. E. Campos[1], Luciano P. Reis[2, 3]

[1]Tecgraf (Computer Graphics Group)/PUC-Rio, Brazil
[2]CENPES (PETROBRAS Research Center), Brazil
[3] Institut National Polytechnique de Lorraine, France

{pozzer,abraposo,ismael,joaoluiz}@tecgraf.puc-rio.br,
luciano.reis@petrobras.com.br

Abstract. *In this work we present CSVTool, a videoconferencing tool implemented in Java that can be easily integrated to different applications. With simple operation, it provides the user means of information exchange, aiming to reduce the barriers imposed by applications with limited or no collaboration support. This tool was designed to be extensible, flexible, platform-independent, allowing a transparent flow of information among the different users with their distinct models and projects. We show the results of its integration with NetGocad, a tool designed for the collaborative construction of earth-models for application in geosciences.*

1. Introduction

Collaborative environments are becoming increasingly more frequent and necessary, mainly in large companies, where the cooperation among specialists in different areas (many times geographically distributed) becomes necessary for the solution of problems. Moreover, many other activities, such as trainings and meetings, may be facilitated and have their costs reduced by the use of collaboration tools.

The difficulties in creating such collaborative environments in industry can be analyzed in two dimensions. In the first one, there is the software diversity that specialists are forced to use to accomplish their tasks in a reasonable time. In the other dimension, there is the necessity of involving specialists in different areas, which causes additional difficulties.

The scenario described above is highly heterogeneous, being composed of not only geographically distributed teams, but also teams of specialists in different areas using different software and different representations for the same artifacts.

There are several ways to treat those problems. Our current approach to start tackling them is to develop a videoconferencing tool that can be easily integrated to different applications, with simple operation, which provides the user means of information exchange, aiming to reduce the barriers imposed by applications with limited or no collaboration support. This tool must be extensible, flexible, platform-independent, allowing a transparent flow of information among the different users with their distinct models and projects.

This work is motivated by the necessity of effective solutions for collaboration of PETROBRAS, a large Brazilian governmental oil & gas company. The necessity of collaboration is especially acute in the field of Computer Graphics, whose techniques such as scientific visualization and three-dimensional geometric modeling have been revolutionizing the industry. The possibility of visualizing and manipulating virtual models in the computer has completely changed the professional's way of working, notably for the geologists, geophysicists and engineers.

The CSVTool (Collaboration Supported by Video), presented in this paper, is a videoconferencing tool developed to tackle those collaboration problems, especially (although not exclusively) devoted to be used in Computer Graphics settings.

In the following section we discuss some collaboration models. Then, in section 3, we present the formal description of the problem, showing the requisites that the solutions must satisfy at different collaboration levels. In sections 4 and 5 some videoconferencing tools are discussed. The CSVTool is presented in section 6, and a case study of its use is discussed in section 7. Conclusions are presented in section 8.

2. Collaboration Models

For a better understanding of the problem, some collaboration models are going to be discussed. The collaboration levels that are going to be presented in section 3 are based on two models that analyze collaborative activities in several aspects: the 3C model (communication, coordination, cooperation) and the model that divides groupware in collaboration-aware and collaboration-unaware applications.

2.1. The 3C Model

In order to work collaboratively, people need to share information (communication). Communication, although vital, is not enough; "it takes shared space to create shared understandings" [Schrage 1995]. This notion of shared workspace (including user awareness, shared objects, etc.) is called cooperation. To cooperate, however, people need to work harmoniously, avoiding conflicting or repetitive actions (coordination). These aspects (communication, cooperation and coordination) constitute a threesome frequently associated with collaboration [Ellis et al. 1991], [Fuks et al. 2002].

Collaborative applications, according to the 3C model, are composed of tools providing one or more of the three functionalities described above. Another central aspect of the 3C model is the notion of user awareness, which is defined as the way users perceive other participants of the collaboration and what they are doing, without direct communication between them [Dourish and Bellotti 1992]. Awareness elements are essential for the collaboration flow, because they enable the user to build his/her own work context and to coordinate his/her activities with those of the others. Therefore, user awareness may be considered the fourth element of the 3C model, which is deeply related to communication, coordination and cooperation.

2.2. Collaboration-aware and Collaboration-unaware Applications

The applications available for computer-supported collaboration can be divided into two categories, depending on how the support for collaboration is related to the application implementation: they can be collaboration-aware or collaboration-unaware applications [Reinhard et al. 1994].

Collaboration-unaware applications are originally developed to be single user applications, but may be used collaboratively by means of an external support system. This external support system may be an application sharing system, such as Microsoft NetMeeting, or a GUI event multiplexing system. The difference is that in the first case, only one user is running the application whose windows' contents are broadcast to all connected users. All users' inputs are gathered and serialized, to be received by the application as if coming from a single user. In the second case, all users are running an instance of the application, with a special layer between the application's GUI and its event handler. This layer broadcasts all GUI events to connected users and interprets all received remote events as if the local user generated them. In both cases the applications do not explicitly support collaboration; they are implemented as single user applications [Tietze 2001].

Collaboration-aware applications, on the other hand, are specially developed for the collaborative work. They constitute distributed systems (centralized or replicated) that are aware of the communication channels among the distributed instances of the application, the information exchange among them, the number of connected users and their role in the collaboration, the coordination policies, among other aspects of communication, coordination and cooperation.

3. Problem Definition

The nature of the problem, as already mentioned, is highly heterogeneous, regarding not only the geographical distribution, but also the software platforms and the specialty levels of participants.

In order to better situate possible solutions, it is necessary to define the adequate collaboration level. The model presented in [Santos et al. 2002], defines hierarchical levels for collaboration scenarios (Figure 1). At each level, different collaboration degrees are presumed.

At level 0, no support for collaboration is defined. At level 1, called video-based awareness, a higher degree of communication is achieved with integrated audio and videoconferencing system. At this level, the collaboration scenario is not complete, since the remote user is not able to interact with the application. At level 2, a degree of cooperation and coordination is possible, given the user capabilities to interact with the remote workspace. Typical applications at this level use the client-server paradigm to exchange information. Level 3 gives collaboration support to applications that were originally developed to be single user, and do not provide explicit support for that. Applications at level 4 are similar to level 3, but the support for collaboration is provided by distributed applications especially developed for that purpose. At level 5, a framework for interoperability among different applications is also supported.

It is important to mention that the lower levels, 1 and 2, though having poorer collaborative resources, are easier to implement and, in some cases, are the only viable solutions due to the available infrastructure and/or budget constraints. Moreover, in some cases where the most important tools used in the environment are commercial software with non-extensible functionalities it is not possible to reach the higher collaboration levels, which require intrusive interventions in the software.

This group of scenarios covers a large spectrum of solutions for many problems. As a generic solution, the level 1 of the hierarchy presents solutions using videoconferencing systems. In this scenario, the concept of awareness is introduced as a mechanism to reinforce communication and cooperation among the users. This solution is adequate for virtual meetings and distance training where, by means of audio and video resources, specialists in a given application or in a specific knowledge domain may solve problems faced by remotely located colleagues, as usually happens in many industrial installations of large engineering companies such as oil platforms, refineries, process plants, etc. For this situation, session management resources were designed for CSVTool and will be discussed in section 6.1.

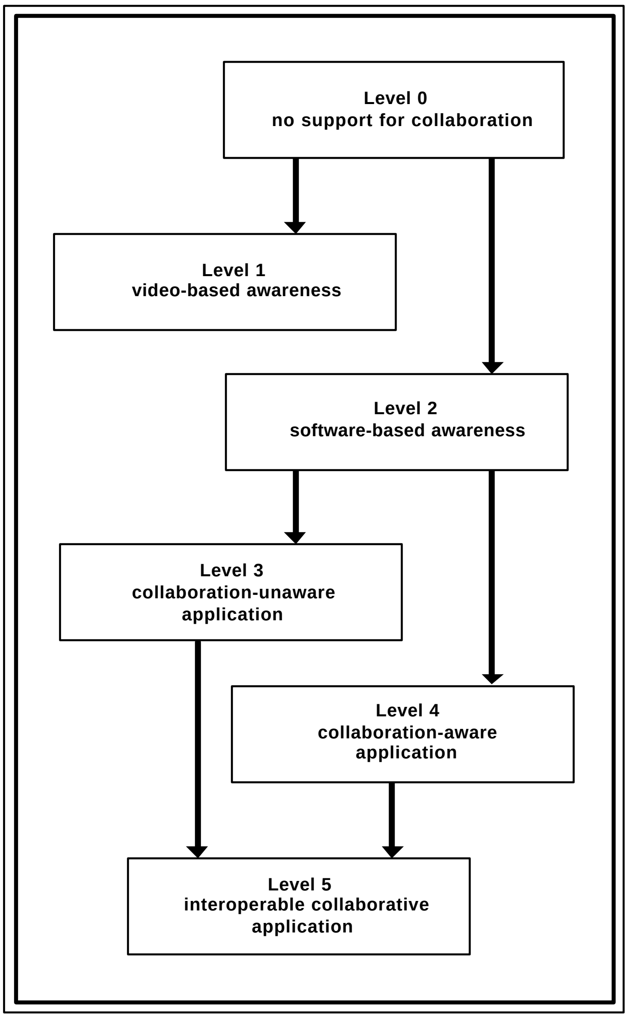

Figure 1: Hierarchy of collaborative scenarios.

With the project evolution, it was noted that CSVTool could be refined to help solving specific problems of both collaboration-unaware and collaboration-aware applications. There are still a large number of operational applications with limited or no support to collaboration. The idea of creating a dedicated version of CSVTool, which could be easily integrated to another application, showed even more interesting than the use of the system as a conventional videoconferencing tool. In this scenario, CSVTool is used as an auxiliary tool for applications at levels 2, 3 and 4 of the above hierarchy, creating an external communication channel, which is almost transparent to the host application. This solution, besides being generic, accelerates the process of the application's evolution toward collaboration, since it requires a reduced programming effort in the original application. This mode of CSVTool will be presented in section 6.2.

So, the CSVTool was designed for two distinct scenarios, providing different functionalities for each of them. The CSVTool was then implemented having in mind both operation modes: as an independent videoconferencing tool (level 1) and as a communication tool integrated to a collaborative application (levels 2, 3 or 4).

4. Related Tools

There are currently a number of packages and applications that offer support to videoconferencing over the internet. All of them offer resources for video capture, coding, transmission and exhibition of audio and video streams. In the context of this project, due to the fact that we need to offer additional communication support for already collaborative applications, which require some interventions in the software, some aspects became crucial for the choice of the software tool to be used, namely, adaptability, flexibility and software architecture. Three tools were studied: NetMeeting, VIC and JMF.

NetMeeting is an audio and videoconferencing tool that enables the communication between two persons, connected via Internet. The video communication of a larger number of users is only possible using a MCU (Multipoint Control Unity), which may be very expensive and, therefore, not viable in many situations. Other resources available are a text chat tool and a whiteboard that allows the real-time collaboration via graphic information. These resources enable more than two users participating in a session. More advanced resources include the application sharing during a conference and the remote desktop, which enables a computer to be operated from a remote location [Microsoft 2003].

Despite the presented resources, NetMeeting is restricted to the MS Windows platform, and has the severe limitation of enabling only two persons communicating via audio and video. The closed architecture also hinders its coupling to other applications to which we need to add communication resources. It also hinders adaptations and modifications to support other services that would be necessary.

Another package that was studied and tested was the VIC (Video Conference Tool), developed by the Network Research Group at the Lawrence Berkeley National Laboratory in collaboration with the University of California, Berkeley [VIC 2003]. It is open source, implemented in C++, multiplatform, and freely distributed. It offers support to various users and enables multicast transmission, which requires the MBONE (multicast backbone) to operate through the Internet. Despite having all the characteristics necessary to this project, the VIC source code was written in low level, hindering the modifications and adaptations needed in the project.

An alternative functionally similar to VIC, but implemented in Java is the JMF (Java Media Framework) [Sun 2003]. This API has a source code in high level, abstracting codecs and transmission protocols details. This facilitates its understanding and, as consequence, the implementation of complex modules. Since JMF was adopted in this work, it will be detailed in the next session.

5. JMF (Java Media Framework)

The JMF [Sun 2003] is an API for incorporating time-based media into Java applications. It is extensible and allows JMF plug-ins to support additional media types or to perform custom processing and rendering.

For the time being, the current API version is 2.1.1e, which provides support for capturing, processing, storing, presenting, transmitting, controlling the type of processing that is performed during playback and performing custom processing on media data streams. The main features of this API are:

- Support capturing media data;
- Enable the development of media streaming and conferencing applications in Java;
- Enable advanced developers and technology providers to implement custom solutions based on the existing API and integrate new features with the existing framework;
- Provide access to raw media data;
- Enable the development of custom, downloadable demultiplexers, codecs, effects processors, multiplexers, and renderers;
- Be platform-independent and easy to program.

In order to support transmission of real-time media data streams, the JFM RTP API is defined, providing support to RTP (Real-Time Transport Protocol). RTP enables the transmission and reception of real-time media streams across the network. RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. RTP does not address resource reservation and does not guarantee quality-of-service for real-time services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multicast networks, and to provide minimal control and identification functionality. RTP and RTCP are designed to be independent of the underlying transport and network layers [RTP 2003].

An application built over JMF has a dataflow very similar to a pipeline. Figure 2 illustrates this process. Arrows indicate dataflow.

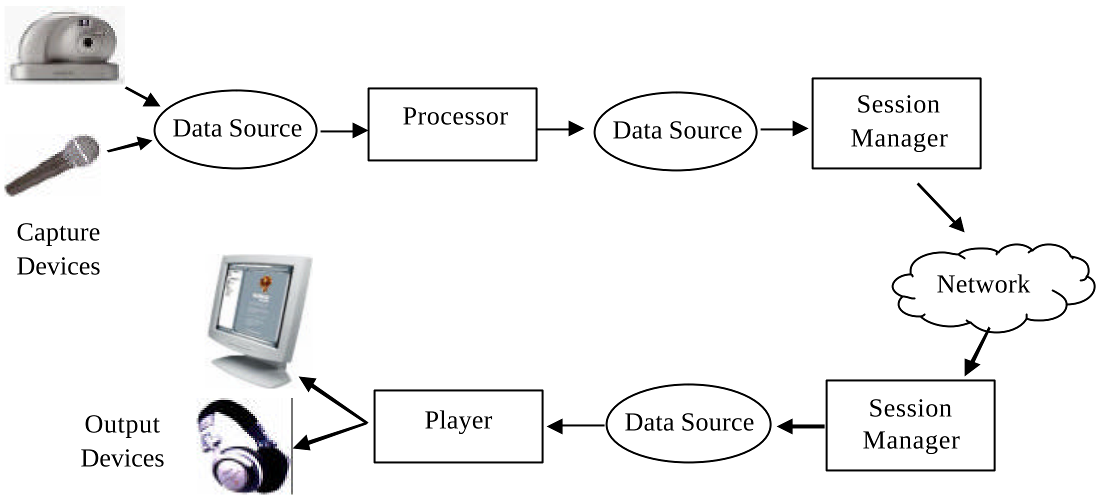

Figure 2: Pipeline for capturing, processing, transmitting and presenting time-based media.

In the first part of the dataflow, a video conferencing application captures live audio and/or video, processes and transmits them across a network using a separate RTP session for each media type. Incoming RTP streams can be played locally, stored to a file, or both. There is also an associated RTP session manager for each incoming media type. A separate player is used for each stream received by the session manager.

Capture is the input phase of the standard media processing. A capture device can be a microphone or a video capture card. The format of a captured media stream depends on the processing performed by the capture device. Some devices deliver raw, uncompressed data, while others might deliver compressed data. Video compression is the most CPU demanding process in the pipeline. To specify the data format and rate, capture controls are also available.

Players process an input stream of media data, delivered by a DataSource, and render it at a precise time, on a device that support the media being presented, like video or sound cards. JMF defines DataSources as data abstraction that encapsulates the media stream much like a video tape. To manage the transfer of media-content a DataSource encapsulates both the location of media, protocol and software used to deliver the media.

Processors are very similar to players. A Processor is just a specialized type of Player that provides control over the processing that is performed on the input media stream, like effects, mixing, encoding, and compositing in real-time. In addition to rendering media data to presentation devices, a Processor can output media data through a DataSource. Processors can use codecs to perform data encoding and decoding, and set quality compression. The greater the compression, the greater the CPU usage and latency during presentation. Typically, effect filters are applied to uncompressed data, and modify the track data in some way.

A SessionManager is used to coordinate a RTP session. It keeps track of the session participants and the streams that are being transmitted. It also allows defining methods that enable applications to initialize and start participating in a session, remove individual streams created by the application, and close the entire session.

6. CSVTool (Collaboration Supported by Video)

The initial conception for CSVTool was to be a videoconferencing tool to connect PETROBRAS visualization rooms, which are rooms equipped with large displays and stereo projectors for visual applications. It can also be used to connect visualization rooms with any other user, placed on any desktop or mobile computer, connected to a network. The operation mode of CSVTool is defined during the startup, by means of specific command line arguments. In this paper we refer to Mode 1 when the tool is operating as a standalone videoconferencing tool, and Mode 2 when integrated to a collaborative application, to provide additional communication resources.

Actually, the tool is developed to support collaborative work among different platforms and users, and both operation modes are implemented on a single Java application. These modes are detailed as follows:

- Mode 1: As a standalone videoconferencing tool, a single CSVTool server may be used to manage several conferences. This mode of operation also offers resources for conference scheduling and configuration, such as start time, topic, conference coordinator and allowed participants. There may be public and private sessions. The former are opened to every client that connects to the conference server, while the second has a specific list of participants.

Additionally, it is also possible to specify the degree of participation, defining if a participant may send audio and/or video to the conference.

- Mode 2: This mode of operation has a simplified user interface, offering only the essential visual components for a videoconference. It does not have, for example, resources for conference scheduling and configuration. The goal is to enable easy utilization, initialization and high adaptability and coupling to applications with collaboration resources in a distributed environment. After the group initialization, which is realized by the host application, the video streams exchange among the participants is automatically started by CSVTool.

Independently of the operation mode, the CSVTool is divided into two modules, the server and the client. The server is an independent module, responsible for the management of the participants and the scheduled videoconferences. It operates in a transparent way regardless of the client execution mode (mode 1 or 2), exchanging the same set of messages in both cases. The core of the client module, which handles the streams is the same for both operation modes, and is showed in Figure 2.

The existence of a server (centralized or local) is vital to the conference management. The information exchanged among clients, except the audio and video streams, is handled by the server, which broadcasts it to the participants that must receive that information. The most common messages are addition/removal of participants, which implies the addition or removal of input and output streams to all conference participants that have permission for that. It is important to reinforce that the server is not prone to traffic overburden because it does not receive the “heavy traffic” (the streams), which are transmitted directly between the clients.

The server/client communication is implemented in CORBA, and the communication among clients, for the streams transmission is made via RTP (Real-Time Protocol). CORBA was chosen for the client/server communication because it is platform independent and widely used. Moreover, it is extensible and allows easy integration with other distributed applications.

In the following sections the main characteristics and differences between the two operation modes of CSVTool are presented.

6.1 Mode 1 – Standalone Tool

In mode 1, clients connect to a fixed central server, whose localization should be known. This server accepts videoconference scheduling and keeps the list of all connected users, as well as the list of active videoconferences, with their respective participants.

The server notifies the participants of a videoconference when there is an alteration in the session group (for example, a new participant is connected), redefining the RTP transmissions to the new group configuration. A diagram of this structure is presented in Figure 3, which also shows the two kinds of data transmission (CORBA and RTP).

When a new client connects to the server, it receives a list of the conferences it is allowed to take part in (public or private), as well as the information about the users current logged in the system. Any server data alteration is forwarded to all connected users. The conference coordinator, which is the one who created it or someone assigned by him/her, is the only person capable of starting and finalizing a conference.

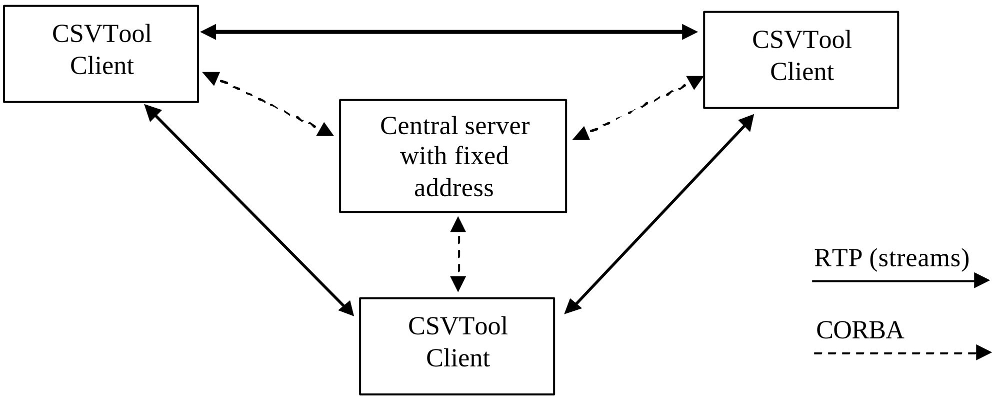

Figure 3: Communication structure between the CSVTool client and server in the standalone operation mode (mode 1).

The interaction project of this operation mode of CSVTool was elaborated based on the proposal of [Barbosa et al. 2002], which defends a human-computer interaction (HCI) design approach based on models and user-centered.

6.2 Mode 2 – Integration Features

This operation mode was developed to be easily coupled to applications with some collaboration resources, as a means to add or extend audiovisual communication, as shown in Figure 4. The coupling process requires little code modification in the original application (see section 7). It must be clear that the CSVTool in this operation mode creates a videoconference session over a collaborative session already taking place. Each of these sessions is executed in an independent application. If the host application does not have distributed resources, CSVTool must be used in mode 1, without making any code alteration in that application.

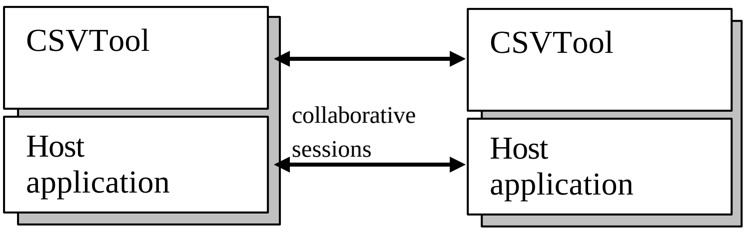

Figure 4: CSVTool integration schema, showing that the collaborative sessions run in independent applications.

The host application should be responsible for initializing the videoconference session after the establishment of the distributed group. Assuming that there is a group of people using the application collaboratively, the idea of the integration is to create an automatic videoconference session for this group, without the necessity of manual configuration by the coordinator of a private session including the same participants, and without requiring their subsequent login, as happens in mode 1.

Independent of the session creation, the coordinator role is also valid for operation mode 2, since someone in the group must start the videoconference, which is realized as unobtrusively as possible.

The adopted solution consists in using the distributed infrastructure of the session taking place in the host application. Initially, a local server is created in the coordinator's machine. Once each participant's copy of the host application has the information about the other session participants, the host application communication resources are used to inform the others that one member of the group wants to start a CSVTool conference. In this initial communication, the address of the coordinator's machine and the server port created for the videoconference session are sent to the participants. By receiving this message, each connected user starts a client instance of CSVTool, which automatically connects to the server created in the coordinator machine. This server is already initialized with a public session, with a specific configuration to attend the group.

The current implementation does not offer any security schema to avoid the participation of unauthorized users, i.e., users that does not belong to the collaboration group defined by the host application. Although each CSVTool conference is created in a variable machine (i.e., in the of the member that decided to start the videoconference) using a random port, this is not enough to guarantee private and security conferences.

The host application must be able to make the CSVTool initialization via system call (Figure 5). This initialization is realized in two steps:

- Initialization of the local server: For this initialization, it is necessary that the host application define the port and the coordinator of the videoconference that will be created. In a videoconference in this operation mode, there is a server dedicated to a single conference;
- Initialization of the clients: For this step, each participant must receive the address and the port of the CSVTool server, as well as the user login.

Once the client and server modules of CSVTool are initialized, they automatically execute the initialization of the default videoconference, already registered in the created server. Initially, the client connects to the server, informing the user login. It then receives the notification of the registered videoconference and the port addresses for audio and video transmissions. The client includes itself in the videoconference and queries the user about the capture devices she/he wants to use. When there is more than one user logged in the server, they start to send streams to each other. When a new participant enters the session of the host application she/he will be automatically added to the CSVTool conference (the host application is responsible to send this information to the CSVTool server).

In this mode of operation, the user interface of CSVTool is simplified, once there is no necessity of configurations (such as session scheduling and definition of participants and their permissions). The goal of this operation mode is to enhance the communication capacity of applications that already offer some form of collaboration, in a manner that is relatively unobtrusive to the users and requires only a simple adaptation of the host application.

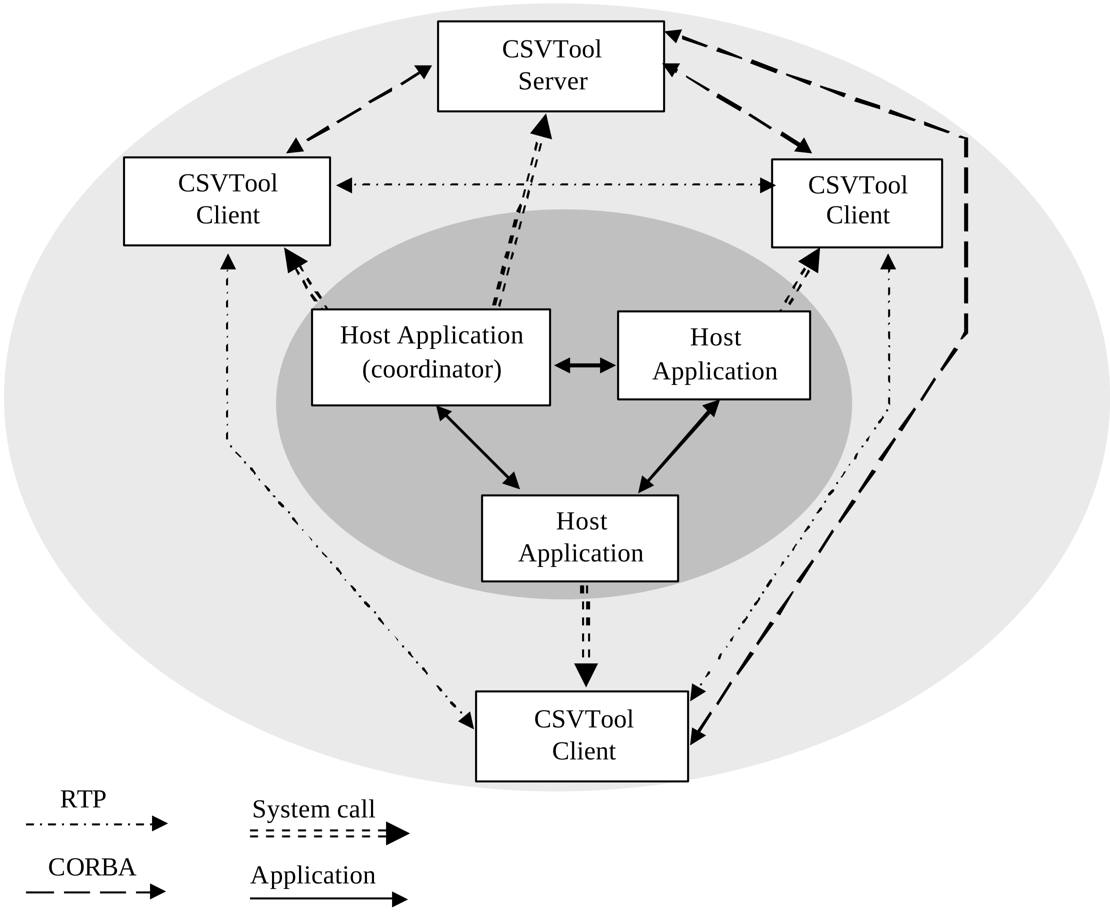

Figure 5: CSVTool server/client communication structure when integrated to a host application.

7. Case Study (NetGocad)

Gocad (Geological Objects Computer Aided Design) is a CAD software [Gocad 2003a], originally developed by an international research consortium [Gocad 2003b], that allows the construction of earth-models for application in geology, geophysics and reservoir engineering. It provides a flexible development framework to facilitate the creation of specialized plug-ins, which can be dynamically loaded at runtime inside the application shell.

NetGocad [Reis et al. 2001] is a collaborative plug-in for Gocad that allows distributed synchronous collaborative visualization and modeling. It allows the creation of heterogeneous collaboration sessions, in which participants may use different versions of a core CAD application, configured with specific functionality and user interfaces (desktop, virtual reality). Communication, coordination and cooperation among multiple users are defined by the NetGocad system architecture, which allows explicit passing of controls among participants, including observer's movement control, model edition commands, 3D cursor, annotations and model transfer. For each type of information, separate communications channels are provided.

In order to complement the collaborative resources of NetGocad and to test CSVTool functionalities in mode 2, CSVTool was coupled to NetGocad by means of a plug-in that offers a graphic interface where the initialization parameters of CSVTool client and server are defined. The NetGocad interface, as well as that of its plug-in, is shown in Figure 6.

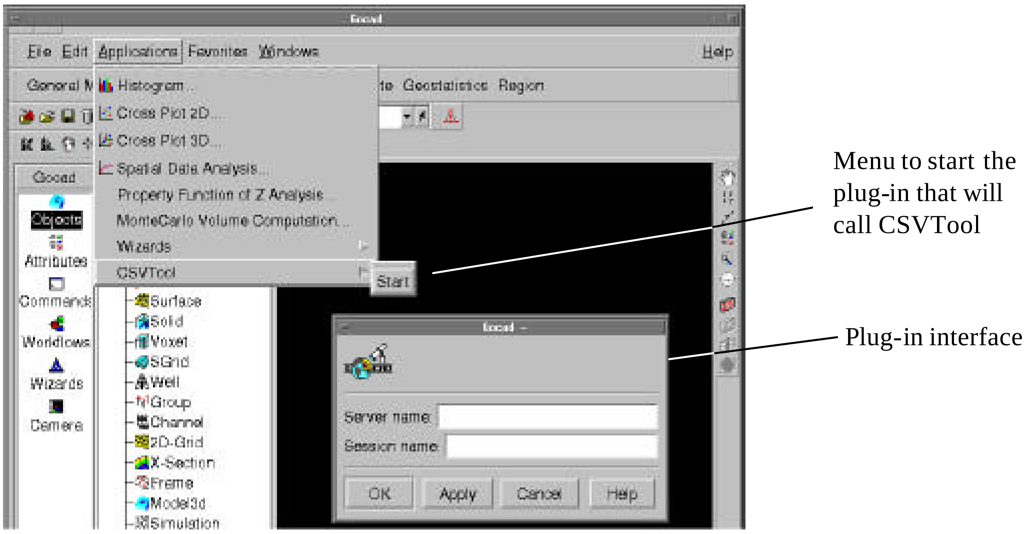

Figure 6: NetGocad plug-in interface for CSVTool initialization.

Once the initialization information is completed, the CSVTool client is initialized and each user must setup the capture devices to be used (Figure 7). The CSVTool client interface is presented in Figure 8. It is composed of a scrollable area where the received streams are displayed. The locally captured video, which is sent to the other participants, is shown in a separate window (capture monitor). This window is helpful for the local user to adjust the correct position and focus of his/her camera.

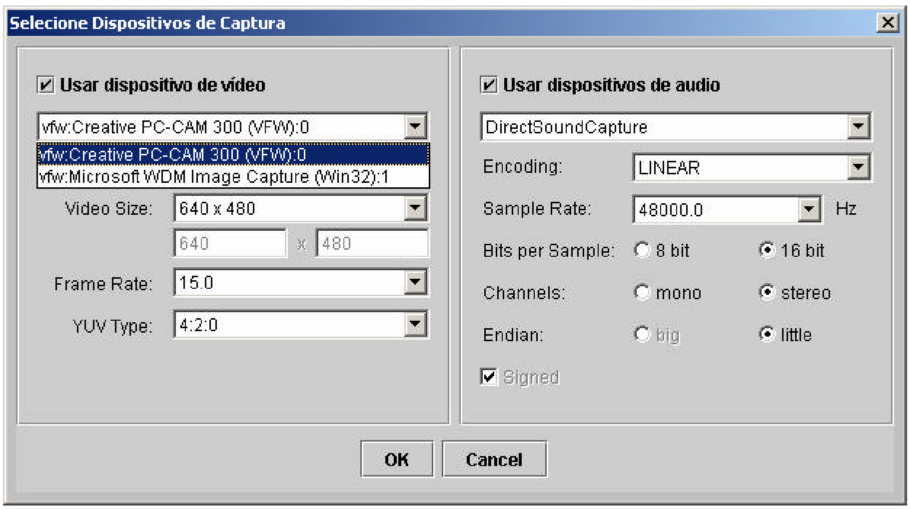

Figure 7: CSVTool interface for the selection of capture devices.

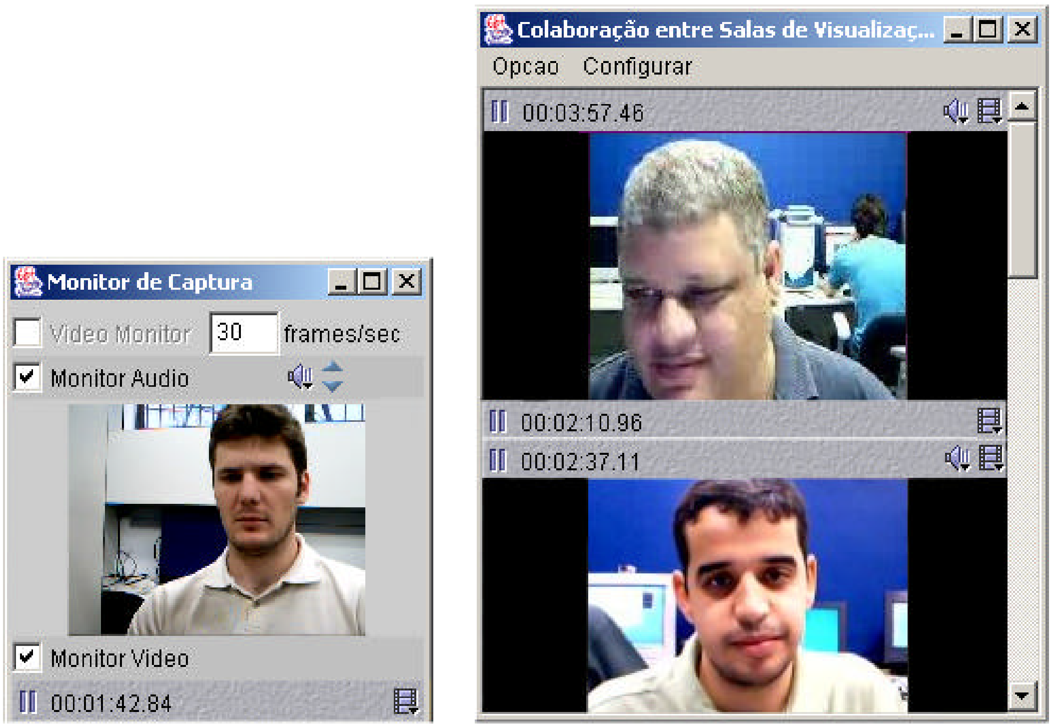

Figure 8: Screenshot of CSVTool capture monitor and scrollable main interface.

Besides its conventional use for communication, the videoconferencing resources can also be used to show the operation of the NetGocad user interface to remote partners, what is very useful for training and consulting activities, since NetGocad does not share interface events. This can be done, for example, in a visualization room, with a camera focusing on the projected screen of NetGocad. During collaboration a lot of knowledge is transferred just by observation of somebody else's work (especially in naive users training). We think that the combination of audio and video information with NetGocad will improve the usability of the Gocad software, which is known to be a very complex and sophisticated software. Further experiments shall be conducted in the future to verify this intuition.

8. Conclusions

In this paper we presented a tool to provide or add collaborative resources to applications in heterogeneous scenarios, where the available collaboration level and the data volume and types may be very variable.

The main characteristic of CSVTool is its easy integration with distributed applications. In order to make this integration viable, the source code of the host application needs to be slightly adapted. However, the cost of this integration is much less than that of adding native resources. Moreover, the form of using the tool is as simple as if it were present in the host application, and not being executed in parallel to it.

Even not providing program sharing resources within host applications, audio and video resources are very useful as an additional means of collaboration. In the case of NetGocad, only the resource sharing (cooperation) and the native concurrency control (coordination) are not enough for a full collaboration, once the tool does not offer communication channels among the users.

Another important feature of video-based collaboration is that the information transmitted is always composed of audio and video streams, independently of the content and the data volume manipulated by the host application. For this reason, the adaptation to new applications, as well as modifications in the host application will not affect its integration with CSVTool.

It is also worth noting that NetGocad is a multiplatform application. Therefore, making CSVTool multiplatform was an essential requisite for the integration.

The CSVTool architecture presents a rapid and cheap solution for a problem that may become very complex, depending on the data volume and the interaction level that one wishes to have in the collaborative application.

As next steps in CSVTool development, it will be necessary to treat some problems, mainly related to the JMF limitation in the detection and manipulation of multiple video capture devices. In PC platforms (Windows and Linux), we succeeded in the detection of various different camera models, but the result was not satisfactory for SUN and SGI platforms. Another issue is that CSVTool was designed with the intention of allowing a single user/machine to have more than one camera, for example, one focusing the user and another focusing the projection screen. In order to allow the simultaneous use of more than one camera in a single PC, the above mentioned JMF limitation must be overcome.

Acknowledgments

The research in collaborative applications at Tecgraf/PUC-Rio is mainly supported by the AmbVirt project, sponsored by FINEP/PETROBRAS. Special thanks to Prof. Marcelo Gattass, head of Tecgraf.

Alberto Raposo is financed by individual grant awarded by the Brazilian National Research Council (CNPq), process nr. 305015/02-8.

Ismael Santos is currently working as a visiting researcher at IMK/FhG, Bonn, Germany.

References

Barbosa, S. D. J., de Souza, C. S., de Paula, M. G. and Silveira, M. S. (2002) “Modelo de Interação como Ponte entre o Modelo de Tarefas e a Especificação de Interface”, Proc. of V Symposium on Human Factors in Computer Systems (IHC 2002), p. 27-39.

Dourish, P. and Bellotti, V. (1992) “Awareness and Coordination in Shared Workspaces”, Proc. of the Conf. on Computer Supported Cooperative Work (CSCW´92), p. 107-114.

Ellis, C. A., Gibbs, S. J. and Rein, G. L. (1991) “Groupware: Some Issues and Experiences”, Communications of the ACM, 34(1), p. 38-58.

Fuks, H., Raposo, A. B. and Gerosa, M. A. (2002) “Engineering Groupware for E-Business”, First Seminar on Advanced Research in Electronic Business (EBR), p. 78-84.

Gocad (2003a) Earth Modeling Solutions: Earth Decision Sciences. Available at: http://www.earthdecision.com

Gocad (2003b) Research Consortium. Available at: http://www.ensg.inpl-nancy.fr/GOCAD

Microsoft (2003) Windows NetMeeting home page. Available at http://www.microsoft.com/windows/netmeeting/

Reinhard, W. et al. (1994) "CSCW Tools: Concepts and Architectures", IEEE Computer, 27(5), p. 28-36.

Reis, L. P., Bosquet, F. and Paul, J. C. (2001) "Toward collaborative geomodeling", 21st Gocad Meeting. Available at http://www.ensg.u-nancy.fr/GOCAD/meetings/ Nancy2001/ index.html.

RTP (2003) RTP, Real–Time Protocol. Available at: www.networksorcery.com/enp/protocol/rtp.htm

Santos, I. H. F., Raposo, A. B. and Gattass, M. (2002) "Finding Solutions for Effective Collaboration in a Heterogeneous Industrial Scenario". 7th International Conference on Computer Supported Cooperative Work in Design (CSCWD), p.74-79.

Schrage, M. (1995) No more Teams! Mastering the Dynamics of Creative Collaboration, Currency Doubleday, New York.

Sun Microsystems (2003) Java Media Framework Home Page. Available at: http://java.sun.com/products/java-media/jmf/

Tietze, D. A. (2001) "A Framework for Developing Component-based Co-operative Applications", Dr.-Ing. Dissertation, Technischen Universität Darmstadt.

VIC (2003) Videoconferencing Tool. Available at: http://www-mice.cs.ucl.ac.uk/ multimedia/software/vic/