Distributed Cooperative Visualization

K.W. Brodlie† D.A. Duce‡ J.R. Gallop‡and J.D. Wood†

†School of Computer Studies, University of Leeds, Leeds LS2 9JT, UK
‡Department for Computation and Information, Rutherford Appleton Laboratory, Chilton, Didcot, OX11 0QX, UK

Abstract
Visualization is essentially a collaborative activity, widely used in many scientific and engineering disciplines. Visualization may be used to convey insight into phenomena that are well-understood, or to present new data with a view to finding new patterns of meaning and new phenomena. Visualization is a powerful tool in presentations (lectures, seminars, papers etc) and in discussions between colleagues. We are seeing a growth in the use of video conferencing to facilitate meetings between participants in geographically separate locations, both specialized facilities (video conference rooms) using dedicated communications channels (ISDN, ATM etc) and desktop video conferencing using the Internet and multicast (Mbone) communications. Distributed cooperative visualization aims to enhance the video conferencing environment (usually the desktop environment) with access to visualization facilities. At the most basic level, pre-generated visualizations may be shared through a shared whiteboard tool. Richer approaches enable users to share control of the kind of visualization generated and the parameters used in the generation. The World Wide Web provides a basis for asynchronous cooperative working and there are experimental extensions in the direction of cooperative browsing. VRML provides the basis for sharing 3D graphics over the Internet. We look at ways in which VRML is being used in visualization, to generate models which may be browsed by participants in a session. Asynchronous collaboration becomes possible by recording the details of the visualization created by one participant, and making this available to subsequent 'visitors' to the Web site. A multiuser audit trail emerges. Developments on multi-user VRML worlds also have potential applications in visualization, providing a form of synchronous collaboration. This STAR reviews the state of the art in these areas, draws out common threads in these diverse approaches and looks at strengths, weaknesses and opportunities for further development in this field.

1. Motivation

Rogowitz[52] has written "Visualization is the process of mapping numerical values into perceptual dimensions". The use of visual imagery to convey scientific insight and truth is not a new phenomenon. Descartes (quoted by Collins[17]) wrote "imagination or visualization, and in particular the use of diagrams, has a crucial role to play in scientific investigation". More recently interest in visualization was focused by the NSF Panel Report on Visualization in Scientific Computing[45]. Their definition of visualization is interesting:

> "Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to *observe* their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights. ... Richard Hamming observed many years ago that 'the purpose of [scientific] computing is insight, not numbers'. ... The goal of visualization is to leverage existing scientific methods by providing new scientific insight through visual methods."

The report highlights the need for scientists to learn to visually communicate with one another. "Much of modern science cannot be expressed in print. DNA sequences, molecular models, medical imaging scans, brain maps, simulated flight through a terrain, simulations of fluid flow, and so on, all need to be communicated visually." Visualization is a medium of communication.

Annex A to the report presents "A glimpse of the future: using visualization tools to solve problems".

> "Consider the following application of the future.

An airplane designer wants to design an airfoil and test it by simulating airflows over it ... Using animation, the engineer can see the shape (external and/or internal) of the airfoil, the turbulence in the boundary layer air flowing over it ... The designer can interact ... in two ways: *interactively* changing the angle of the 3D display in real time or *steering* the simulation. The designer can change the shape of the wing in real time, the speed at which the wing is flying through the air, or the altitude and hence the characteristics of the air ... The designer simulates the new design in real time, alters it in real time, and steers a simulation - all by using visualization techniques. The engineer also saves millions of dollars by not building early prototypes to check the integrity of the design."

Reading this description, one forms the mental image of the designer seated in front of a workstation, interacting with the emerging airfoil design through the rich medium of visualization. Visualization is seen as a component of interaction, a medium of communication, between designer and workstation. The image is that of the *lone* designer, the single user at the single workstation.

However, much of modern science and engineering involves more than one person. Much (one is tempted to say the overwhelming majority) of design, research and development is not the work of one individual in isolation. It is the work of small groups of people, to large teams of people, each with their characteristic skills and expertise, making their contribution to the overall endeavour.

During the 1980s computer networking became widespread in many organisations, and led to the new discipline of Computer Supported Cooperative Working (CSCW) which gathers together researchers interested in how people work together, and how computers and related technologies affect the behaviour of groups of people. CSCW systems started to emerge. Such systems aim to provide support for group working. CSCW systems typically provide audio and video communication channels between participants in a cooperative session with the addition of groupware tools such as shared text editors, shared whiteboard, shared drawing tools, etc.[29] Given the significance of visualization as a medium of communication in a wide range of contexts, the question naturally arises, how is visualization used within group working and how can this be supported in the CSCW system?

The use of visualization in collaborative working might involve a group of people sitting around a meeting table discussing hardcopy output, or viewing a video. It might involve a group of people clustered around a workstation, with one person in the 'driving seat', discussing a visualization, perhaps making suggestions as to how the visualization could be changed in order to draw out other features in the data, for example by changing a colour map or using a different technique to present the data. Participants might take it in turns to 'drive' the visualization system, each working with their own particular data sets. It might involve exchanging visualizations by email, followed by email discussion, or dissemination to a wider community through the World Wide Web. It might also involve the cooperative use of a visualization system as a tool within a CSCW session.

Is visualization any different to other media that may be used in cooperative working? In some senses the answer to this question is no. There are many issues that visualization has in common with other media, for example, text. Control of a visualization system raises similar issues to shared control of a text editor. Who has control? How do participants know who has control? How is control passed between participants? There are some senses in which visualization is different to other media. Visualizations are typically generated by a pipeline or network of processing steps. This raises the possibility of sharing data between participants at different points in the processing, which may lead to a useful tradeoff between data volume and local processing capability.

Bergeron [6] in his introduction to a panel session at Visualization '93 argues that the goals of visualization can be divided into three categories *descriptive visualization*, *analytical visualization* and *exploratory visualization*. Descriptive visualization is used when the phenomenon represented in the data is known, but the user needs to present a clear visual verification of this phenomenon (usually to others). Analytical visualization (directed search) is the process we follow when we know what we are looking for in the data; visualization helps to determine whether it is there. Exploratory visualization (undirected search) is necessary when we do not know what we are looking for; visualization may help us to understand the nature of the data by demonstrating patterns in that data. We can imagine scenarios based around each of these categories.

Descriptive visualization is commonly used in teaching, for example in quantum chemistry visualization is used to convey an understanding of electron density distribution in simple molecules. In computational fluid dynamics, visualization (usually time dependent visualization or animation) is used to convey an understanding of the behaviour of fluid flows. The visualization is typically designed and prepared by the teacher and presented to the students.

Analytical visualization might involve a group of users discussing the interpretation of a data set, for example the results of a run of an oceanographic simulation model.

Exploratory visualization might involve comparing data sets from different sources, for example in geodesy comparing results from bathymetry (sea depth information) with geoid residuals computed from satellite altimetry data (information about the earth's gravitational field) looking for correlations and differences between them.

Comparison between data from different sources is often a fundamental ingredient of collaboration. In the geosciences, for example, it is now realized that much insight is to be gained by sharing data, comparing different kinds of data gathered from different instruments, for example sea height measurements, surface temperature, ocean depth, whereas in the past researchers would concentrate on data from a single instrument which they would almost jealously guard. A researcher's scientific capital was in both the data and the methods used to analyse it. Nowadays the trend is towards sharing and comparison. There seems to be relatively little attention paid to the use of visualization to enable comparison, see for example Pagendarm and Post[48], but this is nevertheless a topic to which attention needs to be paid in systems aiming to support cooperative working.

In the late 1980s, Modular Visualization Environments (MVEs) started to appear, the earliest examples being apE and the first version of AVS. MVEs provide a set of building blocks which perform functions such as reading data, generation of visualizations such as contouring and rendering. MVEs typically provide a visual editor with which to construct applications, by linking together a set of building blocks, and it is perhaps the power of this visualization programming metaphor that has made MVEs so popular today. The modular building blocks may be (but are not necessarily) implemented as separate processes. When this is done, this provides a natural way in which such systems may become distributed systems. Examples of current systems of this kind include AVS[43], Khoros[82], IBM Data Explorer[1], and IRIS Explorer[24]. A very good overview of this whole area is contained in a special issue of *Computer Graphics*[13].

In the early 1990s, the advent of the World Wide Web led to another approach to visualization applications, a client-server approach in which the visualization required is defined through the client, computed, and returned to the client for presentation. The concept of applets provides a mechanism by which part of the visualization computation may be down-loaded to the client and executed client-side. Such approaches are termed *web-based visualization*.

It is useful to distinguish at this stage between three terms:

1. *distributed visualization*: This involves collaboration at the system level. It is interesting that the current batch of MVEs have all been designed with this aim: it is possible to place modules on different computers. Of course this is most useful when it is a computationally intense visualization task, when some modules may usefully be located on a supercomputer, others locally on a workstation. One can also see simple web-based visualization in this class. Although several computers may be involved in the computation, such a distributed visualization system is still a single-user system. Working in a distributed environment does not by itself imply working in cooperation with other users.
2. *cooperative visualization*: Visualization is often a cooperative activity; several people may work together to interpret a visualization. This is collaboration at the human level. It is interesting that the current MVEs did *not* have this as a design requirement, and until recently were all single-user systems. Similarly web-based visualization systems have been single-user. Cooperation is achieved by humans clustering around a single workstation, around a Responsive Workbench device or in a CAVE, for discussion, or through some means outside the visualization system (for example, sending visualization output to a collaborator for comment).
3. *distributed cooperative visualization*: This brings the two concepts together, allowing collaboration at both the system and human level. We shall be able in this STAR to give examples to show that both MVEs and web-based visualization can be extended to this class. A distributed cooperative visualization toolset should allow its users, geographically distributed, not only to run remote resources, but to share images, and possibly also to interact and cooperate, across a network, in the intermediate steps which lead to the creation of the final output.

The next section of this paper presents a structure for the field of distributed cooperative visualization, including both a people view and a machine view. Section 3 looks at environments for cooperative visualization. Sections 4 and 5 look at approaches to supporting collaboration in a wide range of visualization systems. Section 6 considers web-based visualization. Section 7 looks to systems appearing on the horizon which combine visualization and virtual reality. Section 8 looks at some broader categories of collaborative systems, distributed collaboratories, and section 4.7 draws the paper to a close with a discussion of open issues and future directions.

2. A Structure for the Field

2.1. People View - Types of CSCW

CSCW involves both people and machines. In the CSCW literature, Applegate's place-time matrix is a widely cited classification scheme for cooperative working[4]. This scheme, shown in Figure 1, focuses on the people involved in CSCW and the way in which the types of involvement may be classified.

Two dimensions are used: time and place. Members of a CSCW group may be located at the same place or a different place, and be present in a CSCW session at the same time or at different times. There is thus the notion that a session may extend in time, and not all participants need be present at the same time.

The same time, same place box corresponds to all members of the group being present in the same place at the same time. Examples of systems to support such forms of working

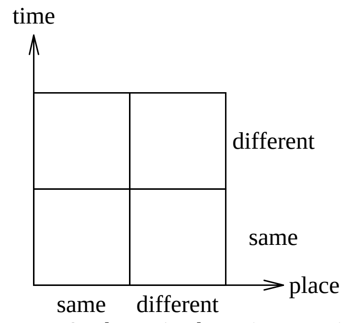

Figure 1: *Applegate's place-time matrix*

include conference rooms equipped with individual workstations to support decision making processes. In visualization a group of users clustered around a single workstation, around a Responsive Workbench or in a CAVE, would fall into this box.

Collaboration that involves exchange of letters, faxes, emails, between members of a group, falls into the different time, different place box. There is a notion of a group session, the group working on a common problem over an extended period of time. There is a shared history of working contained in the trace of letters, faxes, emails, etc. exchanged between group members. The World Wide Web typically falls into this box, though developments such as cooperative web browsing[33, 42, 51, 53] are extending the web to other forms of working.

Video conferencing falls into the different place, same time box. Members of the group are co-present in time, but at different physical locations. Video conferencing may involve a specially equipped video conferencing suite, with multiple cameras etc. at one extreme, or may be based on a suitably equipped workstation per participant. The latter approach can be characterised as collaboration as an extension of the normal working environment. This notion is taken further in the work of Fuch's group at UNC Chapel Hill[25]. which is investigating the use of cameras and sophisticated display technology to assign a region of each office to collaboration, so that one's collaborators are brought into the office in an even more direct way than through workstation-based video conferencing. However, collaborative working is not just video conferencing. There is a need to share information other than through audio and video channels, and a need to share applications used in the creation, analysis and presentation of information. "Groupware" falls into this category. Much work in distributed cooperative visualization aims to support this type of collaboration.

In any one collaboration, it is likely that several different types of collaboration will be used, for example, formal face-to-face meetings, coupled with different time - different place styles of working between meetings, coupled with informal same time, different place sessions. This raises issues about seamless transitions between different types of working, and the organization of group memory so that it is equally accessible from different types of meeting. Xerox PARC, for example, have worked extensively in this area [7].

2.2. Machine View - Types of CSCW

The previous section looked at a way in which the human component of CSCW can be classified. In this section we consider how the computer component may be classified.

Prior to the late 1980s, it is true to say that visualizations were generated by specific programs, perhaps written using high level library routines (such as contouring routines) and a graphics package for generating output. To change the type of visualization generated or visualization technique used required the user either to modify the program or move to a different program. In the mid 1980s, the advent of modular visualization environments (MVEs) provided a more convenient framework within which to develop and use visualization applications. MVEs allow a user to build a visualization application by connecting together predefined modular components, using a visual editor. Most such systems also provide interfaces that enable the user to add new modules. MVEs also allowed different modules to run on different computers, thus introducing a distribution dimension.

In the early 1990s, the advent of the World Wide Web provided a new model for cooperative working and also a convenient way in which visualization services can be delivered. With the web, the emphasis moves away from the user constructing or selecting a particular application to generate a particular type of visualization, to a more declarative approach in which the user describes the type of visualization required and the data set to be visualized, and the "system" returns a visualization of that type. Web-based systems are intrinsically distributed systems as a consequence of the client-server nature of the browser web-server architecture. Applets, written in languages such as Java, provide another mechanism through which a computation may be distributed between client-machine and server-machine.

For the purposes of this STAR, it is convenient to classify the machine view in the dimensions MVE-based and web-based, as shown in Figure 2. Examples of all four combinations will be seen later in this paper.

Adding the people dimension introduces a further classification of the combined human-user system, shown in Table 1, this is based on the number of humans (1 or N) and the number of computers (1 or M). For this paper, the interesting combinations are (1,M), (N,1) and (N,M).

2.3. Problem Size

Visualization techniques typically deal with datasets ranging from kilobytes to terabytes. When dealing with very

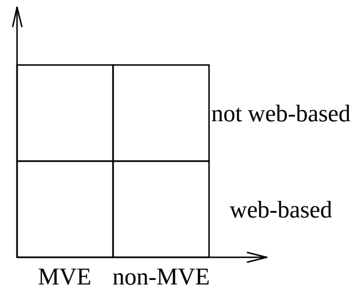

Figure 2: *Machine-view classification*

Computers / Humans	1	M
1		Distributed Visualization
N	Cooperative Visualization	Distributed Cooperative Visualization

Table 1: *Classification of combined human-computer system*

large data sets, the organization of the data set, for efficient access approaches paramount importance, and the issues of space/time tradeoffs in sharing visualizations are very important. Shipping terabytes of data across a network, or copying terabyte data sets are not the answers and so attention focues on extracting regions of interest, finding the appropriate level at which to transmit data and so forth. The system architecture tends to be founded on considerations of data management.

For smaller datasets these problems are still present, but in a less acute form and more flexibility is available in the choice of architecture. It needs to be said that small-scale visualization is just as important as large-scale visualization. The insight gained is not a function of problem size alone.

2.4. Models

2.4.1. Different-time, Different-place, Web-based

In this section we will focus on approaches to visualization over the World Wide Web. The web can be used as a way of disseminating information, but it also has a role in analysis. Users may specify a data set from a web browser, select a visualization of that data set, and - somehow - a visualization is returned to the browser.

Brodlie[8] has developed a reference model for this area, based on three "players": the user, the visualization service provider and the data provider.

The User

This may be the specialist scientist or engineer, or just a member of the general public. The range of skill and experience can be quite varied - in all cases, we can assume that they are familiar with a web browser; in certain cases (the experienced researcher), they may be familiar with an existing visualization system. The computing facility available on the desk can also vary - in all cases, we assume it is powefrful enough to run a web browser with VRML plug-in and Java interpreter; in certain cases, it may be a high performance workstation capable of running a visualization system.

The Visualization Service Provider

This player hosts a web page which manages the visualization facility. This will allow the user to specify the location of the data, and the means by which they will be visualized. There are three levels of service:

- *full service*: in which the visualization is entirely created by the service provider, and returned as an image or 3D model to the user;
- *software delivery*: in whch the software to create the visualization is downloaded to the user, to be executed by them;
- *data only*: in which the visualization software is assumed already resident with the user, and only data need to be delivered.

The Data Provider

This player supplies the data. There are three possibilities:

- *Independent agency*: this would be some organization providing a service by collecting and publishing data of particular interest (for example, the UK Atomic Energy Authority collect air quality statistics and publish these on the web);
- *The User*: the user and data provider may be the same person; this would typically be the case in the traditional use of visualization by a scientist to analyse their own data;
- *The Visualization Service Provider*: here the roles of data and visualization service are combined - for example, when a data provider saw visualization as an essential means of interpreting the data.

From this analysis, we can develop a simple schematic model of scientific visualization over the web, as illustrated in Figure 3.

The three players are placed at different locations on the Internet: the User is the client, the Visualization Service Provider acts as the server; the Data Provider is logically at a third location, although as noted above, may be linked to either of the other two players. The User runs a web browser; the Visualization Service Provider hosts a web page for the service. The parameters in the model are:

- *visualization software*: which can be provided by the visualization service, or by the user;

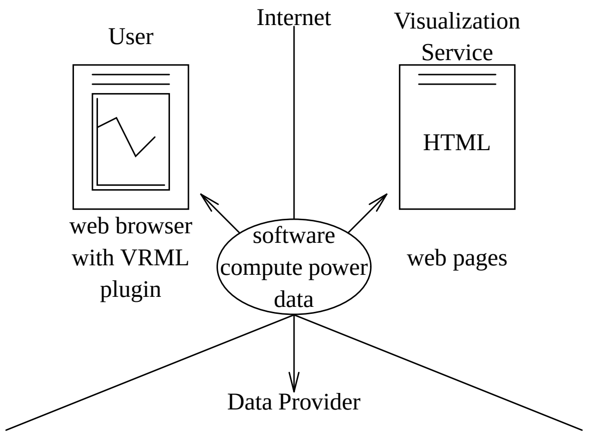

Figure 3: *A reference model for visualization over the web*

- *compute power*: which again can be provided by the service or the user;
- *data*: which may be provided by an independent agency, user or visualization service.

The area of web-based visualization is examined in section 6.

2.4.2. Same-time, Different-place

Same-time, different-place may use the same tools as different-time, different-place collaboration, but may also use tools which give users the ability to share control over the way in which a visualization is generated, whilst retaining some degree of local control over the way in which the visualization is presented in the environment local to each user.

Two approaches may be cited as extrema:

- *Application sharing*: in this approach, a visualization system is run in one location (we factor out the possiblility that the system might be a distributed visualization system). Sharing takes place at the user interface level. Output from the system is distributed to all participants. Input may be generated by any participant and sent to the system. The visualization system is not aware that it is being used in a collaborative session. Examples of this approach include Shared-X and to an extent Microsoft's NetMeeting. Because of the low level at which sharing is handled, this approach tends to be operating system and window system dependent.
- *Output sharing*: this is a weak form of sharing; output from the visualization system is distributed to all participants, for example through a shared whiteboard system. There is an overlap with the approaches to visualization over the World Wide Web, if one considers output from the visualization service provider being sent to multiple destinations. There are a number of projects addressing multicast web browsing, though not specifically concerned with visualization[33, 42, 51, 53].

MVEs offer intermediate possibilities for sharing control and distributing output. These possibilities arise because modules may be introduced which provide support for collaborative working, thus it becomes possible to pass data and control information between instances of the visualization systems run by different users in the collaborative session.

In this way it is also possible to see cooperative visualization as an extension of ordinary visualization. Collaborators may be introduced into a visualization session seemlessly. Collaboration becomes an extension to the normal working environment, not a replacement for the normal environment.

Wood, Wright and Brodlie[77, 76] and Duce et al.[21] have described reference models for this type of collaborative working. The Wood, Wright and Brodlie model is an extension of the familiar Haber and McNabb visualization model. Haber and McNabb [30] describe visualization in terms of the sequential composition of three types of processes, originally termed data enrichment, visualization mapping and rendering, but now (using terminology due to Upson[71]) referred to as Filter, Map and Render.

The extension of the model to encompass cooperation is accomplished with the introduction (potentially) at each stage of the pipeline of intermediate import and export points for control information and data. The model in its most general form is shown in Figure 4.

F denotes the filter transformation, M denotes the visualization mapping and R denotes rendering. The horizontal arrows represent the progression of raw data through the transformation pipeline, emerging as an image. Control information can be imported from or exported to another pipeline at each stage. This is represented by the process parameters symbol. Similarly, data can be imported from or exported to another pipeline at each stage. This is denoted by the vertical arrows branching from the horizontal arrows between each processing stage.

The key concepts captured by the notation are:

- The generation of a visualization may be described by a three stage processing pipeline, following the Haber and McNabb model.
- Each processing stage is controlled by a set of parameters.
- A distributed collaborative visualization system can be modelled by a collection of pipelines, each of which represents the processing stages 'owned' by a particular participant. These pipelines may be complete (contain all stages) or partial (contain only some stages, e.g. only the rendering stage). The stages visible to all participants are indicated by a surrounding box.
- Control information may be exported from one pipeline to others in order to synchronize parameter values between pipeline stages.

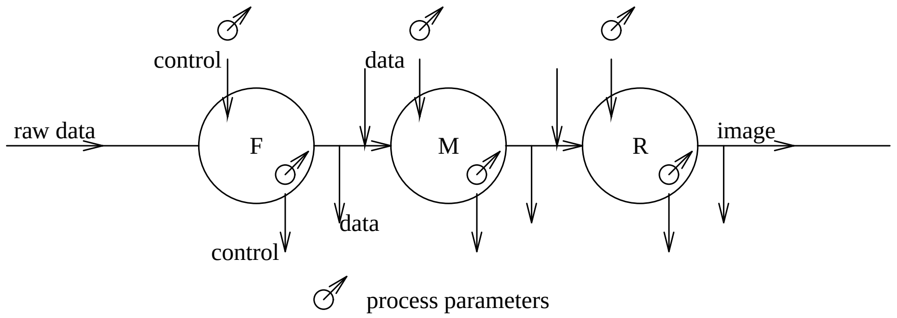

Figure 4: *Wood's model*

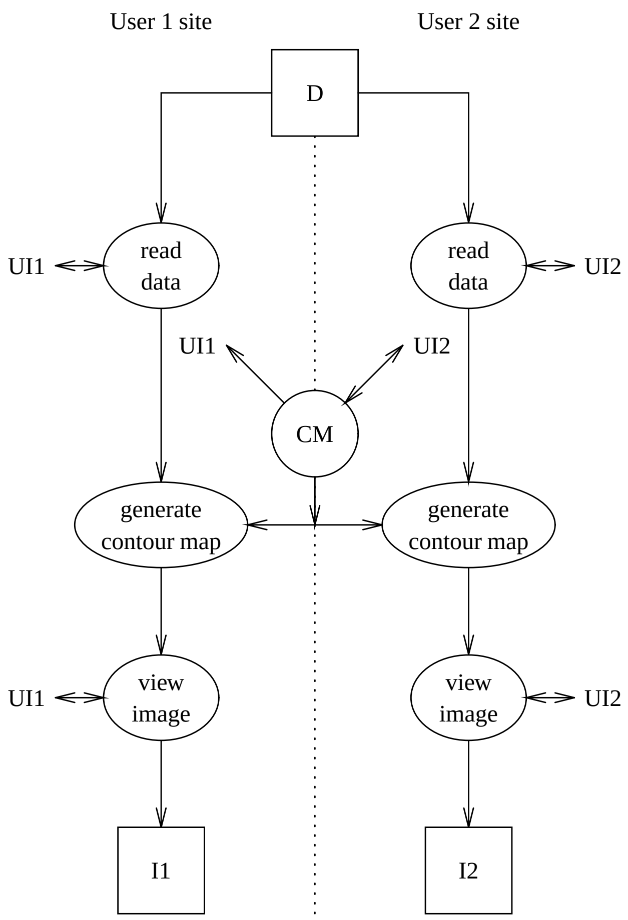

Figure 5: *A representation of a distributed cooperative visualization application in the MANICORAL model*

An application of the model due to Duce et al. which will be referred to as the MANICORAL model (the name of the project in which it was developed) is illustrated in Figure 5.

The model sought to extend the work of Wood, Wright and Brodlie in a number of directions:

- The explicit representation of the interaction mechanism that allows a user to control the parameters of a module, through the introduction of an associated control module encapsulates and simplifies issues related to arbitration between different input sources and dynamic changes in control and arbitration.
- The granularity of transformation processes in the MANICORAL model is not restricted to the filter, map and render granularity in the Haber and McNabb model, though this granularity can be used in the MANICORAL model. The advantages of allowing a wider range of granularity are that it is possible to describe visualization systems, or more specifically visualization applications built using a modular visualization environment, at a range of levels of description, including the 'native' level inherent in a modular visualization environment.
- When more than two users are involved, the diagram in Figure 5 would become 3-dimensional, a set of planar sheets joined along a common edge, each representing the environment of one user and the common edge representing sharing between users. This sheet notation gives a convenient way to represent modules over which each user can exercise a form of control at a moment in time.
- The MANICORAL model identifies the user with whom a particular module is associated, in a rather more explicit way than the Wood, Wright and Brodlie model.

2.5. Analysis Factors for Distributed Cooperative Visualization Systems

In this section, we consider frameworks in which distributed cooperative visualization systems may be compared and contrasted.

Duce et al.[19] used the following factors to compare the approach taken in the MANICORAL project with approaches taken by a number of other systems. The systems compared (including the MANICORAL system) were all based on Modular Visualization Enviornments (MVEs), and MVEs extended in a variety of ways. The factors used were:

- *Base visualization system.* The system into which support for collaborative working is introduced.
- *Level at which data are shared*. Some systems limit the points at which data can be shared, for example between filtering and rendering. Other systems allow data sharing between any pair of modules.
- *Floor control*. The approach taken to managing sharing of control. Some systems adopt a *laissez-faire* approach - the system will accept input from any user at any time and it is up to the users to agree who has control at any time using protocols outside the system. Other systems provide support for passing control between users. Others have a strict master-slaves relationship in which only the master can exercise control.
- *Shared control of module parameters.* Some systems allow control over the parameters of individual modules.
- *Scalability*. In CSCW systems in general, there is an issue of how many users the system, or system architecture can support. Some architectures have inherent limits, others have greater expansion capability.
- *Joining/leaving a collaborative session.* This factor looks at how new users can join existing collaborative sessions. This builds on the notion of collaboration as an extension to the normal working environment and the idea that users should be able to join and leave a CSCW session as they may join and leave other forms of meeting. Clearly there is an issue of how new users acquire the current state of the session.

Wood in his PhD thesis[80] analyses systems according to a number of factors: range of platforms, functionality, participation, system and target user. These factors were used in the evaluation of the COVISA system, developed at Leeds University, and are summarized below.

- Multiple Platforms.
  - In a collaborative session, it is more natural for each individual user to be able to use their own desktop system. There is therefore a requirement to support collaborative systems across a heterogeneous set of workstations, rather than tie a solution to a single platform.
- Functionality.
  - *Exchange of data* - rather than simply being able to share the raw data, the system should be able to share data from any point within a pipeline.
  - *Exchange of parameters* - this would enable, for example, two collaborators to jointly steer the visualization, by sharing control of modules in their (probably) common pipeline.
  - *Exchange of modules* - to allow one user to automatically launch a module into the environment of their collaborators. This should also allow the automatic copying of the parameter set.
  - *Exchange of networks* - this extends the point above to cover a number of modules along with their interconnections. This would allow, for example, a more experienced user to set up a network for a novice, or allow a collaborator to send a fragment of a network to other collaborators complete with parameter settings and connections. This is potentially useful to help collaborators who join late catch up with the current state of the system.
- Participation
  - *Setting up* - The initial setting up of a collaborative

session should be as simple as possible, requiring the least effort on the part of the participant. Ideally, all elements required for setting up should be put in place by an administrator if possible.
  - *Joining/leaving* - It may not be possible for all collaborators to be available at the start of a conference, or to remain until the end. Facilities should exist for users to join and leave at any time.
  - *Automatic launch/connection* - As the collaborative system extends the modular paradigm of dataflow, where modules can be added to the system at any time, it is important to aid users by automating the external connections of the shared elements.
  - *Floor control* - Users should be able to set the type of conference control that they require. This can be used to offer different levels of access to a session to individual users - for example to create a 'See What I'm Showing' style of conference.
  - *Privacy* - In addition to participating in a conference where all elements are shared, users need to work privately while still remaining part of the conference. This is required to support conferences that contain parties with different skill backgrounds. For example, consider the design of an aircraft wing: a materials specialist may wish to look at tensile strength of a cross section, while a flow analyst will be interested in the air flow over that section. Both, however, need to be aware of how a single design change will affect their area of interest and hence need to collaborate. This type of group needs the flexibility to share some parts of the pipeline while having other, domain specific, parts under their own control.
  - *Global View* - The ability to view the network editor of other collaborators is useful to reassure users that they understand what each user is doing. It also improves the collaborative map building process since an expert user can more easily aid a novice. However, this will be more than a simple view of any collaborator's entire pipeline since the ability to have private work contexts may mean that some pipeline elements are not shared.

- System
  - *Performance* - The addition of collaborative elements to the single user system should not lower the overall performance of the tool. Also, shared data objects should be routed as quickly as possible between collaborators.
  - *Reliability* - Data objects passed into the collaborative session should be guaranteed to arrive at the correct output points intact. All participants should also be guaranteed that the data objects they are sharing are identical for all collaborators.
  - *Robustness* - The system should be able to survive the failure of any one part without the entire session collapsing. For example, if one user is suddenly unavailable the rest should be able to continue.

- Target User
  The tools need to be applicable to a broad range of users, with different skill levels in their use of visualization systems. In particular, we need to address two distinct classes of user:
  - *Visualization Programmers* - These would be considered an expert user of the base visualization system. They would be comfortable with dynamically constructing visualization pipelines determined by the current direction of their investigation.
  - *Visualization End User* - These are not expert users of any particular visualization system, yet derive benefit from using tailored visualization applications.

The scheme of Duce et al. is subsumed well by the more general and systematic scheme of Wood. If base visualization system and scalability are added to the system factors, Wood's scheme neatly accommodates the other scheme.

2.6. Architecture

There are fundamentally two approaches to distributed cooperative visualization: the centralized approach and the replicated approach. In the centralized approach a single copy of the visualization application is executed (though in saying this we do not rule out the possibility that the single application is a distributed application). In the replicated approach multiple copies of the visualization application are executed, typically one per participant.

In the centralized approach, the output from the visualization application is distributed to all participants and input control information generated by each participant is sent to the application. The style of operation is thus send control, receive output.

In the replicated approach, the visualization (or parts of the visualization) are computed separately for each user. Synchronization mechanisms are needed to enable all participants to view the same visualization at defined points.

Hybrid architectures in which some parts of the visualization application are centralized whilst others are replicated are also possible. Basing the visualization application on an MVE does not prescribe either approach. There are examples of cooperative MVE-based systems which exhibit both fundamental architectures.

2.7. Summary

This section has laid the foundations for the remainder of this paper. The scope of the systems considered in this review is focussed on the same time, different place and different time, different place, parts of the Applegate classification matrix, though by way of additional motivation, section 3 looks at some examples of systems in the same time, same place, category. Within this limited field, there is a spectrum of interest from loosely-coupled systems based on client-server World Wide Web technology, through to tightly-coupled systems built as modular visualization environments, or extensions to such environments. Although web browsing is commonly thought of as a solo activity, this is not necessarily the case, and when web browsing is thought of as a collaborative activity there is even more reason to include web-based approaches in this survey.

Reference models for both web-based visualization and MVE-based visualizations have been described. The model of web-based visualization concentrates on the players involved: the MVE models take a more system-based approach and concentrate on interfaces at which sharing can take place and representing the kind of sharing that is actually taking place or may potentially take place.

The section concluded with a discussion of a set of factors that can be used as the basis for comparing different MVE-based approaches.

3. Environments for Cooperative Visualization

In this section we briefly describe some examples of environments for the same-time, same-place category of collaborative working in Applegate's time-place matrix. This is not the main focus of this paper, but it is useful background to consider some examples in this category.

The simplest type of same-time, same-place environment is the group of users clustered around an ordinary workstation. The obvious problems with this environment are (a) only one person can 'drive' the system, there is only one set of input devices and (b) different participants inevitably view the display from different directions. These difficulties are faced by more sophisticated approaches to same-time, same-place environments.

Projection systems provide one approach to display for same-time, same-place collaboration for a group of users. For visualization involving 3D scenes, stereo projection is often used, either with red-green glasses or more sophisticated shutter glasses.

The Responsive Workbench developed at GMD[38] is centred on a real bench with a special glass plate as a bench top. Images are projected in stereo onto the glass plate from below. This approach grew out of experiments with a range of device hardware in applications including medical imaging, molecular design, fluid dynamics visualization, autonomous systems and architecture. A head tracking device is attached to the user. Stereo images are viewed through shutter glasses. In multiple user scenarios, the users have to move around the workbench as a group. The position of only one user is tracked, so the perspective view is only correct for one person. Object distortion arising from the single user perspective rendering was found to be a significant drawback.

A variant on the Responsive Workbench which tracks two users and generates correct stereo from two positions has been described[2]. The users wear shutter glasses. Four images are computed, one for each of the four eyes, and displayed in sequence. The system requires very powerful hardware to generate images fast enough. One of the limitations is that image brightness is reduced because each image is displayed for a shorter period of time than in a single user system.

Another approach is the CAVE (CAVE Automatic Virtual Environment), developed at the Electronic Visualization Laboratory, University of Illinois at Chicago [18]. The CAVE is described as a theatre, typically a cube of dimension 3m, made up of three rear-projection screens for walls and a down-projection screen for the floor. Full colour screen images (1280 x 512 stereo) at 120 Hz are projected onto the screen. The CAVE is also equipped with a multispeaker audio capability. A user's head and hands are tracked with electromagnetic sensors. Users have to wear LCD stereo shutter glasses. Groups of people can enter a CAVE and experience the virtual environment, but like the Responsive Workbench, there are limitations caused by interference between users and the fact that the position of only one user is tracked, leading to a collapse of stereo vision.

Lehner and DeFanti[41] have described a collaborative virtual prototyping system developed at GMD and NCSA for the Caterpillar company. The system is designed to allow engineers at Caterpillar in the USA and Belgium to work together on vehicle designs using distributed virtual reality. The system integrates real-time video transmissions into the virtual environment, allowing engineers to see other participants. The paper anticipated trials in which a CAVE at NCSA would be linked via multicast prototcols over IP over an ATM network to a Responsive Workbench at GMD.

4. Systems Based on Modular Visualization Environments

This section discusses some of the work that has been done using MVEs as a basis.

The first group (COVISA, COVISE, MANICORAL and the SDSC collaboration system) takes advantage of the highly visible data flow networks in an MVE application and allows shared control of any module in the network.

The second group (ONERA, NASA Ames work - FAST) does not place such emphasis on visibility of all modules.

In all cases, same-time, different-place cooperation is supported.

4.1. COVISA

- *Who*:
  University of Leeds, funded by the UK EPSRC
- *Based on*: IRIS Explorer from NAG

COVISA exploits an MVE to allow shared control of any parameter and shared data at any visible data connection in the data flow network. It is also possible to launch a module in a collaborator's network editor, or map editor as it is termed in IRIS Explorer. This mechanism also allows an entire map to be shared among the users in the collaboration, thus allowing all users to start from the same basis. Once one user creates a share module, all other users in the session have a companion module automatically available to them. More than two users are possible, and users can leave or join at any time. Users can author their own collaborative modules using a supplied API. Collaborative end-user applications, where the map is hidden from view, can be constructed combining IRIS Explorer's standard facilities for packaging a map into a turnkey system, and the collaborative extensions. Any application domain for which IRIS Explorer can be used is possible, but the COVISA system has been demonstrated with medical and CFD applications.

References[77, 76, 79, 80, 62]

4.2. COVISE

- *Who*: University of Stuttgart, funded in part by the E.U. RACE project PAGEIN.
- *Based on*: The cooperative system is based on the COVISE system itself.

COVISE is designed to accomodate very large datasets, using a shared data space which modules running on all cooperating workstations may access. The intended model is to allow multiple users to visualize the results of a simulation running on a supercompter. One user at a time may exercise control, but control can be changed to another user in the course of a session. The project was primarily concerned with design and simulation of aircraft.

References[39, 57, 75, 74]

4.3. MANICORAL

- *Who*: CLRC RAL, funded by the E.U. Telematics Programme.
- *Based on*: AVS / Express

The MANICORAL project exploits the modularity of an MVE to allow shared control of any parameter at any visible data connection in the data flow network. Each participant runs a copy of AVS/Express. Sharing is introduced by incorporating sharing modules into the networks of the participants who wish to share parameters or data (parameter sharing was fully implemented, data sharing was designed and partially implemented). Sharing modules communicate through a so-called distribution server. Unlike the COVISA system, each user has to explicitly introduce sharing modules to their network, although once the possibility of sharing a parameter is created by one user, any other user may join in. The system has been tested with 3 simultaneous users. The main applications in the project were altimetry and geodesy based on GIS.

References[19, 20, 21]

4.4. SDSC - AVS-based

- *Who*: San Diego Supercomputer Center, funded by the U.S. Naval Oceanographic Office.
- *Based on*: AVS5

The project aims to extend an existing visualization system. It exploits the modularity of an MVE to allow shared control of any parameter and shared data at any visible data connection in the data flow network. The main application in the project is oceanography.

Reference[64]

4.5. ONERA

- *Who*: ONERA, funded as part of the E.U. RACE project PAGEIN.
- *Based on*: AVS5 and IRIS Explorer

The ONERA work aims to extend an existing visualization system. The result of a simulation may be delivered to the collaborative visualization system, which allows the parameters of the visualization modules to be controlled by any of the users in the collaboration. A number of cooperative visualization applications are pre-configured using tcl/tk. The main application is aircraft simulation and images of this work are available in[69].

References[28, 39]

4.6. NASA Ames work - FASTexpeditions and Remote Collaboration in FAST

- *Who*: NASA Ames
- *Based on*: The collaborative work is based on the FAST visualization system. Although not strictly an MVE, it is convenient to discuss it here.

There are two main approaches to cooperation in the FAST work.

FASTexpeditions allows cooperation at different times via the Web. One user sets up a dataset and a FAST script. Recipients can run a FASTexpedition by accessing the script via the Web. The script is run on the recipient's local machine using a local copy of the dataset.

FAST also allows Remote Collaboration by multiple users at the same time.

The primary application of FAST is CFD.

References[5, 60]

4.7. Open Issues/Future Directions

There are a number of open issues raised as a result of experience with MVEs. We focus here mainly on experience gained through projects at Leeds and RAL in which the authors have had a direct involvement.

Introducing sharing. An MVE provides a good framework for constructing a DCV system as others, as well as ourselves, have observed. There is a natural transition from a personal application to a shared application, by introducing sharing modules into the application. However, unless the need for sharing is envisaged by the application designer, it is necessary for edits to be made to the module network (application) in order to introduce sharing. Some systems, notably COVISE, provide a form of shared editing, so that changes made by one participant are reflected in every participant's network. Other systems, notably COVISA and MANICORAL, assume independence of editing, so that each participant is responsible for their own pipeline (which may be different from that of others). The responsibility for connecting the sharing module then lies with each participant, allowing them the flexibility of connecting it where they wish. In the case of COVISA, when a share module is launched by one participant, a companion module is automatically launched in the map editor of every other participant, leaving a simple wiring operation. COVISA also includes an Advisor module which allows one participant to launch a module, or group of modules in every other participant's map editor. This is useful in situations where one participant is more experienced than another. In MANICORAL, use is made of the AVS/Express parameter blocks mechanism to provide a convenient way to introduce sharing. Parameter blocks allow a network to be set up with local control, then by dragging-and-dropping a sharing module onto the parameter block, sharing is enabled. Parameters are given names and once one participant has declared a willingness to share a parameter with a given name, any other participant may join in shared control of that parameter, providing a parameter block with the appropriate name has been included in that participant's network. Thus COVISE, COVISA and MANICORAL are points in a spectrum of possibilities. There may be a need for compromise here and the recognition that production usage and development usage have different requirements.

Differential workstation power. If the workstations used by some participants are significantly less powerful than those used by others, the slower workstations can be swamped by changes generated by the faster workstations. New parameter values are received faster than the workstation can recompute the visualization. A similar problem is created by differential network bandwidth between participants.

User interface integration. Outside the visualization field, some work has been done to integrate and customize the user interfaces of audio-video conferencing tools for particular applications, for example for remote language teaching in the ReLaTe project [12], but these rely on the user interfaces of all the tools being constructed with the same UI technology, in that case Tcl/Tk. Some visualization systems, for example COVISE, integrate the user interface to the audio-video conferencing tools with the interface to the visualization system. Other approaches, such as COVISA and MANICORAL, keep them separate. In practice COVISA has mainly used the SGI InPerson tool which has video-conferencing and a shared whiteboard. MANICORAL is used with the Mbone tools: the audio tool RAT[31], the video tool vic and the whiteboard tool wb. The stance taken on this issue may be dictated purely by practical considerations. It may be impossible to achieve integration because of the way in which the tools are constructed. There might be a difference of approach depending on whether the tools are to be used for well-defined production tasks, or much more informally in a setting where the task is not well-defined and cooperative working is relatively informal and opportunistic as an extension to (not a replacement for) the normal working environment.

Swamped by windows. One of the difficulties of MVE-based approaches is that they tend to generate large numbers of windows. In the MANICORAL system, for example, some attempts have been made to group the control panels for shared parameters together, but these attempts have been hampered by what appear to be limitations of the Express UI toolkit. In a recent demonstration of the MANICORAL system, good use was made of the multiple desktop capability in the window manager used. Different desktops were used to hold firstly visualization output and active input controls, and secondly the network editor. Multiple desktops can provide a useful way to structure activity in a collaborative session. Nevertheless, the management of screen real-estate remains problematical.

Scalability. It seems to us that most examples of distributed cooperative visualization systems have only been exercised with small numbers of users, typically 2 to 4, and are best suited to small group exploratory working. Extension to larger groups (such as might arise in teaching scenarios) in a scalable way remains an open issue, raising protocol questions (such as reliable multicast protocols) as well as distributed system architecture questions.

Asynchronous collaboration. Most attention to date has focused on synchronous collaboration. However MVEs are also used in major simulation applications, where the simulation code is included as a module in the pipeline. Such applications may take hours, even days, to run, and in these cases synchronous collaboration is of limited use. Thus a topic for future work is to study how mechanisms for asynchronous collaboration may be introduced.

Privacy and conference control. The issues of privacy and conference control are complex. In certain collaborative projects, participants may wish to keep some of the data confidential, while still allowing public access to the remainder of the data. The MVE systems such as COVISA and MANICORAL that allow independent editing of the visualization pipeline by each participant, provide a ready solution to the privacy issues: you share what you want to share, you keep private any data you want to keep private.

Floor control is another issue. In COVISA, a social floor control policy is adopted so that participants agree (by video conference communication) who has control at any point in time. MANICORAL takes the same approach, relying on human protocols to mediate control. This has actually proved successful in practice.

The joining and leaving of a session, that is, more general conference control, is a harder issue. In COVISA, a central server keeps a record of the current set of share modules and so they can be dispatched to any latecomer. However, in order for them to make sense of these modules, another participant will have to send an example pipeline showing how to connect them (using the Advisor module mentioned earlier). Finally the pipeline data held on each of the shared modules has to be explicitly sent.

Network services. The systems described in this paper have used a variety of networking infrastructures, from local area networks to the Internet and dedicated broadband ATM connections. From experiments carried out in the MANICORAL project, in which the RAL authors were involved, it became clear that the Mbone and Internet in their current form are inadequate for desktop audio-video conferencing. Our experience of sessions with 8 to 10 participants is that it is common for 2 or 3 sites to experience network difficulties such as high packet loss rates or high variance in packet delivery delay times during any session. Audio quality and reliability are such that participants have to develop human protocols for frequently checking who is present in a session and whether all participants are seeing the same display, i.e. whether all updates have reached them and been processed. In the limited experiments carried out with the JAMES ATM network (in which only two sites were involved), the situation was completely different. The audio quality was consistently high and the need to continually check what was happening elsewhere disappeared. In our view, for distributed cooperative visualization to be viable, a range of network services needs to be available at the desktop to support the reliable delivery of data streams and the near lossless delivery of audio and video streams with low delay variance. Provision of such services is an open issue in multiservice networking.

Introducing a new community to CSCW. We write of our own experience. The greatest difficulties encountered by the users in the MANICORAL project (in the experience of two of the authors) were concerned with networking. In many cases the MANICORAL participants were the first groups at their sites to request access to the Mbone. In some countries and organizations the Mbone is regarded as a service, but in many countries the Mbone is run on a best endeavours or good will basis and consequently it is unreliable and it can be difficult to establish the necessary router connections. Establishing connectivity through the JAMES network also required considerable communications expertise. Ready access to appropriate networking services and expertise is an essential prerequisite.

5. Collaboration in Other Visualization Systems

Although MVEs now dominate the visualization marketplace, there are a number of alternative approaches - some which are new, and some which pre-date MVEs but retain a significant following.

The early approach to visualization was to provide a library of routines (typically coded in Fortran) which a user could include within their own software in order to visualize their data. Examples of this genre include the NAG Graphical Library[9], which is still well-used today even though over twenty years old. A more modern variant is the vtk toolkit of Shroeder et al. [54], providing a C++ class library for visualization.

These systems could in principle be used to provide the basic components of a collaborative system, although we are not aware of any work in this direction. One reason for this is simply the work involved in developing the collaborative framework around the software - it is much easier in the case of MVEs because the existing MVE framework is designed to support distributed working, and as we have seen can be extended to support distributed cooperative working.

However there are some new approaches to visualization, in the form of interactive packages where collaboration has been incorporated as a design aim.

One of the first visualization applications to address collaboration specifically was the *Tempus Fugit* system for CFD visualization, developed by Gerald-Yamasaki[27]. Indeed distributed visualization is also studied carefully in this paper. In *Tempus Fugit*, a partition is identified at the geometry creation stage, so that processing up to that point in the pipeline is carried out in a server process on a supercomputer, and the geometry is then passed to a client process on a graphics workstation for rendering. *Tempus Fugit* has a companion program, called Interview, which extends the system to collaborative working. An Interview client connects both to the *Tempus Fugit* client and to the server: upon request to the *Tempus Fugit* client, the Interview client can receive a list of the geometry objects that client is currently viewing, and these objects can then be downloaded from the server to the Interview client. Both clients then share the same geometry, and are free to interact with it independently - although it is possible to synchronise views if desired.

The Spray rendering system of Pang et al. [49] is a novel approach to visualization, with some similarities to MVEs but also some significant differences. In the Spray system, a user points a 'spraycan' at a set of data: smart particles (*sparts*) are fired into the data with a particular objective (such as 'locate isosurface') and these sparts create geometrical primitives (polygons in the isosurface example). Rather than data flowing through modules as in MVEs, it is more a case of modules flowing through data.

CSpray[50] is a collaborative extension of the Spray system. The active members in a collaboration each have a spray-can, and they jointly create a shared visualization space by 'painting' the data. The people are represented as *eyecons*, which are essentially avatars in the form of an eye-ball. The eyecon shows where the corresponding person is sited in the visualization space. By clicking on an eyecon, another user can move to that person's viewpoint in the visualization space.

Some important issues have been covered in this work. The issue of *privacy* is carefully studied: a collaborator can make their spray-can either private or public - the results of a private spray-can are seen only by that person, whereas the results of a public can are visible to all. *Floor control* is implemented on a per-object basis: a user can request control of a visualization object such as an isosurface from the person who created it, or the person who last manipulated it. An *audit trail* of the session is maintained: this allows latecomers to join in by re-running the session to that point; or a session to be replayed at a later time.

CSpray has been used in a number of application areas, notably environmental science through the REINAS [23] project.

There are two recent systems which should be noted. Both are Java-based, and both include collaboration as a fundamental part. The first is the Sieve system developed by Isenhour et al. [32]. This has some of the flavour an MVE-style system in that data input, filter and visualization modules can be connected in a dataflow pipeline. Collaboration is of the WYSIWIS (What You See Is What I See) style, meaning that all collaborators use the same pipeline; but with 'location-awareness' so that different people can simultaneously edit different parts of the pipeline. The state of a Sieve session is maintained on a central server, so that latecomers can join the current session - and indeed it allows *asynchronous* collaboration, in which one user can deposit the final work of a session, for another person to take over at some later time.

Another new system is the NPAC SciVis[63], developed at Syracuse University. This is designed as a client-server system, with the visualization system as the server, and the data generation process as the client. Data generation may be simple file reading, or it can be a simulation which creates data. The two processes communicate via UNIX sockets. The visualization server is written in Java, and supports a form of collaboration in which users can exchange data and personalized filters (processes which transform data).

6. Web-based Visualization

6.1. Introduction

The World Wide Web has grown from its initial focus as an information repository, to have a much wider impact as a distributed computing environment. This has seen the role of visualization in the Web extend in a similar way. Its early use was for simply *descriptive visualization* (in Bergeron's categorisation) where a prepared visualization of scientific data was placed as an image, or perhaps a VRML model, within a Web page. It is still often used in this way of course, providing a very useful means of publication of results. Most visualization systems now allow VRML as an output option - see Walton[72] for example.

However the visualization community has realised that it is also possible to carry out *analytical* or *exploratory visualization*, where the investigative process itself is carried out in a Web environment. The first work that we are aware of in this area is that by Ang et al. [3], and this has been followed by a number of studies covering a wide spectrum of approaches to web-based visualization.

Here we review the different approaches which have been proposed. We shall use a very broad classification, determined by the location of the visualization processing. Recall the Haber and McNabb reference model, which sees visualization expressed as a sequence of Filter, Map and Render processes. The geometry created from the Map process, and passed to the Render process, provides a convenient means of distinguishing two broad classes of approach. The distinction is between:

- *Client-side visualization*: here the entire visualization execution (Filter, Map and Render) is carried out on the client, perhaps within the Web browser; thus the geometry is created on the client-side.
- *Server-side visualization*: here the Filter and Map processes are carried out on a server - so geometry is created on the server; the Render process, where the 3D geometry is viewed, is carried out normally on the client (but perhaps on the server).

Within each category, we shall see that different flavours of the approach are possible. This is covered in section 6.2.

The Web is fundamentally a cooperative environment and any use of the Web will involve interaction between the publisher of the source material on the server, and the subsequent readership. However it is now becoming possible to think about cooperation between the readers themselves: we therefore look at how the existing approaches to single-person visualization on the Web can be extended to include cooperative working. This is covered in section 6.3.

6.2. Distributed Visualization Over the Web

6.2.1. Client-based Systems

In this category, the complete visualization execution - the Filter, Map and Render processes - is executed on the client. Within this approach, however, there are three styles of working which vary according to how the software is assembled.

At this point, it is convenient to see visualization software in two parts: the *visualization design*; and the *core software*. In the traditional library context, the visualization design is the user program, the core software is the library of subprograms. In the MVE context, the visualization design is the visual program connecting modules into a pipeline, the core software is the set of modules provided with the MVE. The design and core software together form a visualization executable.

We can then allocate Client-based Systems into three separate categories, depending on whether the visualization design, and the core software, are present on the client, or are downloaded from a server.

(a) Visualization Design and Core Software Both Present on Client

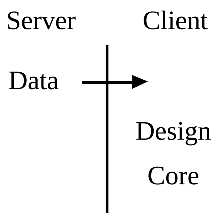

Figure 6: *Client-based : Design and Core on Client.*

In the simplest style, we assume the visualization executable (design and core software) is fully installed on the client machine, and there is no software to download. If the data to be visualized are also on the client, then it does not make sense to involve the Web at all. However it becomes of interest if the data is not local, but rather is available at a URL (perhaps generated by a collaborator, or perhaps a publically available source of data, such as air quality information, or stock market prices). We can then use the visualization system as a 'helper' application, or plug-in, fired up by the browser on receipt of data of a specific MIME-type. See figure 6.

Indeed this was the basic approach taken in the pioneering work of Ang et al. [3]. Their work focussed on volume visualization, and they defined the MIME-type 'hdf/volume' to identify volumetric data in the NCSA Hierarchical Data Format (HDF) representation. On receiving a file of this MIME-type, they were able to invoke the inter-client communication facilities of the Mosaic browser, to fire up the user interface of their volume visualization system, VIS.

Another example of this is the Web version of Vis-5D [66], developed by Hibbard at the University of Wisconsin. Vis-5D is a widely used system for the visualization of meteorolgical data. It is a turnkey system, so the design and core software are closely integrated. The name of the system comes from the type of data files it processes: the 5D are (typically but not necessarily) latitude, longitude, altitude, time and one of pressure, temperature, wind velocity and so on. By convention, these datafiles have extension 'v5d'. The Web server delivering Vis-5D files is set up to assign the MIME-type 'application/vis5d' to files of extension 'v5d'; similar set up is required at the browser side, to associate Vis-5D with this MIME-type. Then, on receipt of a Vis-5D file over the Web, the browser will fire up Vis-5D as helper application. The Vis-5D web pages give examples of its use in this way, for instance to provide daily weather forecast visualizations from data delivered by the US National Weather Service.

An MVE can of course be used in exactly the same way as the Vis-5D system. Module pipelines, suitable for different data types, are prepared in advance and held locally. Each data type has an associated MIME-type. On fetching data of a particular type from a Web server, the browser fires up the appropriate pipeline to process the data. An inhibiting factor to this style of working is the lack of an international standard for visualization data, in the way that VRML has become for geometry data.

Jern has discussed an approach like this[34], based on the AVS/Express visualization system. He discusses the use of visualization plugins to perform data manipulation and rendering locally at the client side. He is also exploring how component frameworks can be used to provide visualization components that can be incorporated into documents in an active way[35].

(b) Visualization Design Downloaded from Server, Core Software Present on Client

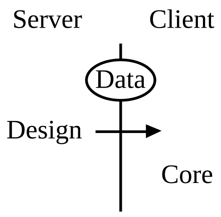

Figure 7: *Client-based : Design Downloaded, Core on Client.*

In the case just described, it was only data that was being transferred across the Web. However, particularly in the case of an MVE, there are more exciting possibilities. As mentioned above, execution of visualization within an MVE can be seen as two distinct processes: first, the creation of a pipeline of modules specifying the particular processing to be carried out - the visualization design; second, the core software - the modules themselves, and the basic infrastructure of the MVE. In this approach we download the design from the server, linking to core software already present on the client - see figure 7.

Thus one can see the MVE as an empty workspace into which a visualization program, that is, a specific pipeline of modules, can be transferred across the Web. Thus a set of 'example', or 'demonstration' pipelines can be held on a server, and associated with a particular MIME type; on being fetched by the browser, the pipeline can be automatically loaded into the empty workspace as the application is launched. This has significant potential for education and training, and is being studied currently by Yeo[81]. The pipeline of modules is then available for use in the normal manner by the user.

Note that this can be combined with the first style of working, where data is read by the visualization system from a URL rather than local file. The example pipelines which are downloaded could themselves include modules that read data from URLs rather than local files - thus enabling both data and 'design' to be delivered from a Web server.

Some visualization systems can be driven by a scripting language. They too can be integrated with Web technology using this general approach. The system is installed on the client, but the script is downloaded over the Web. An example of this approach is the FAST[5] system for CFD visualization; as we shall see later, this script delivery over the Web is used in their collaborative extension.

(c) Visualization Design and Core Software Downloaded from Server

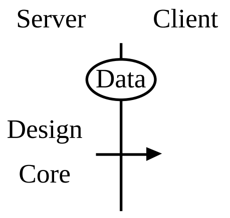

Figure 8: *Client-based : Design and Core Downloaded.*

In this third category, the entire visualization software (design and core software) is downloaded dynamically from the server - see figure 8. The enabling technology is of course Java, and the visualization executable is provided by Java applets embedded in an HTML page.

An early example is described by Michaels and Bailey[46]. VizWiz is a Java applet which creates isosurface, cutting plane or elevation grid visualizations, from data supplied by the user. A difficulty with this general approach is that the data must be held locally on the server which delivers the applet, for security reasons. This is not ideal! It is likely that the data will either be held locally by the user on the client, or in the case of shared public data, at some URL (not necessarily where the applet is). Michaels and Bailey solve the problem for local data by using the Netscape file upload option, which transfers a local file to the server. A major snag is that the data is transferred to the server and back again, just to circumvent security restrictions!

An alternative solution has been proposed by Kee[36] and Stanton[55], allowing data from any URL. The data are fetched temporarily to the server by a Java application (distinct from the visualization applet) which executes on the server and, being an application, is not subject to the security restrictions of applets. However, again there is large data transfer involved.

From a commercial rather than a technical viewpoint, it is of interest to be able to charge users for their use of the applets. Kee and Stanton experimented with a 'credit server' which logs the time an applet is in use.

Another example of this approach is given by Wegenkittl and Groeller, in their FROLIC[73] system. This creates visualizations of dynamical systems, using a variant of Line Integral Convolution and other methods. The systems are defined analytically, and so there is no problem with uploading data.

This section has concentrated on software downloaded dynamically during a session. We note however the Java systems mentioned in section 5 such as SciVis and Sieve, fall into the category of systems which may be downloaded over the Web (being written in Java); but this is more an initial, static download rather than the on-demand type download which is the principle interest here.

6.2.2. Server-based Visualization

In this category, the main visualization processing is carried out on a server and graphical data is transmitted to the client, for viewing within the browser. Again there are different styles, determined by the extent of the rendering delegated to the client: the graphics data may be in the form of a rendered image, requiring only image display software; or it may be in the form of a 3D VRML model, requiring a VRML browser on the client. This Server-based visualization is distinguished from the Client-based approach above by the fact that the geometry - at least - is created on the server.

This approach reduces dramatically the requirements on the client side - in terms of processing power (only rendering is required); in terms of software availability (only image or 3D graphics software needed, and commonplace in browsers anyway, with VRML now an ISO standard for 3D Internet graphics); and in terms of training (only ability to use image or VRML tools required). Nothing comes free of course, and this approach requires the presence of a server with all the ingredients that have been removed from the client: processing power to create the visualization as an image or 3D model;

visualization software itself; and human involvement to create suitable visualization applications. Moreover, if several clients connect at once, then the processing requirement on the server can be quite severe.

(a) Image Display on Client

In this style, the entire visualization processing - filter, map and render - is executed on the server. An image in some recognised format (JPEG, PNG, GIF) is transmitted to the browser.

A nice example is CurVis developed by Theisel et al.[58]. This uses a novel technique of flow visualization, by displaying the curvature of streamlines. The visualization is returned as an image, but the user has control over the format and compression.

(b) 3D Model Rendering on Client

In this style, the rendering is delegated to the browser. VRML is generated on the server and transmitted to the client, allowing the viewer to have control of the rendering process.

An early example of this approach is described by Wood et al.[78], based on IRIS Explorer as the visualization system. The basic principle of the method is as follows. The user enters details on a form to specify the location of the dataset (as a URL say), and a 'recipe' for the style of visualization required. The form is processed by a CGI script on the server, and a visualization server is executed. This invokes IRIS Explorer, using its scripting language (Skm) to define an appropriate pipeline. The output of the pipeline is a VRML file, which the visualization server returns to the browser for viewing.

A demonstrator has been created, to view air quality information in the UK. This data is collected on an hourly basis at a number of locations throughout the UK, and the Atomic Energy Authority (AEA) publish the data on a Web site. The demonstrator allows a user to select the time and location of data, and the pollutant to be analysed. The approach can in principle be extended to the presentation of many different types of public service information - for example, road traffic data, weather forecast details, and so on.

A similar idea has been developed by Pagendarm and Trapp[70, 47] in their Vis-a-Web system. There are two significant differences however: first, the forms interface is replaced by a very flexible approach using Java to provide the GUI; second, the visualization server makes use of specialist software, including modules from the HighEnd system. A novel feature is the fact that software from different sources can be used to make a *single* visualization - VRML files from different sources can be combined before transfer to the browser.

A further system has been developed by Lefer[40]. Again the basic principle is the same as proposed by Wood et al., but here the underlying software is based on the VTK toolkit of Shroeder et al.[54].

Indeed the VTK authors themselves now include instructions on their web site[68] explaining how to build a VRML visualization service. The procedure is simple: first install VTK on the server; then create a VTK program to read user parameter values and to produce the required visualization (using 'vtkVRMLExporter' to write the VRML output), saving the executable in a cgi-bin directory; next, create an HTML form to allow a user to enter parameters to the program and trigger execution of the program.

Treinish[61] , in a paper on visualization design for operational weather forecasting, describes how IBM Data Explorer can be used in a Web environment. He envisages a partitioning of Data Explorer into a client-server system, with a Java-based applet on the client interacting with Data Explorer on a server.

All the above approaches follow a common model: the server process accepts requests from a form interface, or Java program; it executes the visualization on the server; and returns VRML to the client.

6.2.3. Review of Web-based Visualization

We have seen therefore that Web-based visualization can be divided into two classes: Client-based systems, and Server-based systems. Within each class, there are different approaches: Client-based systems vary according to whether the visualization design and core software are downloaded from a server, or are assumed already installed on the client; Server-based systems vary according to whether an image, or a 3D VRML file, is returned to the client.

In figure 9, we try to show this taxonomy of web based visualization in a diagram. The front face of the structure relates to where the visualization processes of Filter, Map, Render and Image are executed - either on Client or Server; and the side face relates to where the software emanates from - again either Client or Server. Alongside we mention the various examples that have been discussed earlier.

A number of issues may help to resolve which approach is best in a particular situation. If high interactivity with the dataset is required, then a Client-based approach may be best, allowing data, visualization and interaction to occur locally. Note however that the Java-based Client systems may involve significant start-up costs in data transfer for security reasons.

For the novice user, Server-based systems offer an attraction. The installation, licensing and other start-up costs can be borne by the provider of the remote visualization service. In addition a simpler user interface can be provided, to hide detail from the beginner. Server-based systems can also be useful in an Intranet situation, where an organisation can set up a visualization service tailored to a particular group of scientists.

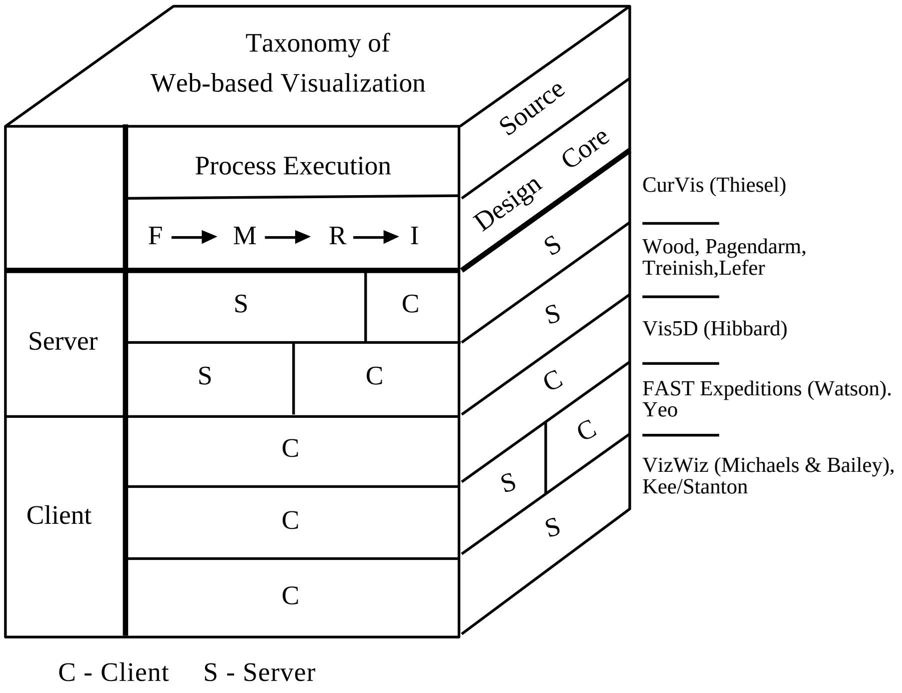

Figure 9: *Taxonomy of Web Based Visualization.*

The choice of approach will also depend on the relative processing power of the client and server, in comparison with the size of the visualization task. The size of the dataset likewise needs to be taken into account, relative to the available network bandwidth. If data is being transferred from a remote location, it may well be sensible to apply the filter process before the data transfer, in order to economise on bandwidth.

6.3. Cooperative Web-based Visualization

In this section we shall explore how the approaches described above can be extended to cooperative working. Some of these are untried ideas because there has been relatively little research to date, but we may expect this to be a significant area for new work over the next few years.

6.3.1. Cooperative Client-based Systems

We saw earlier that these fall into three different classes, according to the software that has to be downloaded over the Web. It is convenient to follow the same classification for cooperative visualization.

(a) Visualization Design and Core Software Both Present on Client

This is the case where the launch of the visualization system on the client is triggered by delivery of data of specific MIME-type. The system will consist of visualization design plus core software, tightly integrated in the case of a turnkey system, or more loosely coupled in the case of an MVE or graphics toolkit/library. In the MVE case, it is possible for a collaborative pipeline to be invoked. Thus, if a number of different users all fetch data over the Web at the same time, then they can begin a cooperative session using one of the extensions to MVEs described earlier (for example, COVISA or MANICORAL). Thus this approach extends almost trivially to cooperative visualization.

(b) Visualization Design Downloaded from Server, Core Software Present on Client

There is some real practical experience of this approach, through the NASA work on FASTexpeditions. As explained earlier, the FAST system can be driven by a script: in a FASTexpedition, the visualization tasks carried out by one user (the ‘pilot’) can be recorded internally as a script of commands; these commands can be transferred from pilot to another user, a ‘passenger’, so they can see the same visualization. The roles of pilot and passenger can be interchanged between participants. FASTexpeditions [60] have been used in an educational context.

In the context of an MVE, this case corresponds to the download of a pipeline of modules. The extension to cooperative working is straightforward in principle: for example, in the COVISA extension of IRIS Explorer described earlier, it simply is a case of downloading a pipeline which includes the special data sharing modules.

(c) Visualization Design and Core Software Downloaded from Server

This is the case typified by the downloading of software dynamically in the form of Java applets (such as the SDSC VizWiz system). We are not aware of any work to extend these to cooperative working.

6.3.2. Cooperative Server-based Systems

Almost all the research in cooperative visualization has focussed on the same-time, different-place quadrant of the Applegate matrix. However it is not always convenient to collaborate at the same time: perhaps the collaborators themselves have different schedules, or perhaps the processes themselves do not fit a synchronous style of working (for example, when the visualization data is output from a large simulation process running over several days). Here asynchronous collaboration is more appropriate.

This has been studied in the context of Server-based systems by Wood[79]. He extends the air quality demonstrator described in 6.2.2 to allow a number of collaborators to investigate data over a period of time. It works as follows. A first user creates a visualization using the form front-end to the IRIS Explorer system running on the server. If an interesting visualization results, they may choose to store the parameters which defined that visualization. Note that these parameters completely define the visualization, and of course require much less storage than saving, for example, the VRML output.

Now consider another user in the collaborating group. On fetching the HTML page, they can also gain access to the data saved by the earlier user. They can use this as a starting point for their own study of the data: they can add their own textual comment on the visualization, or they may choose to change some of the defining parameters, creating a new visualization. If this new visualization is of interest, its defining parameters may be saved for later use by other collaborators. This sequence of stored definitions effectively forms a tree of exploratory visualizations, built up over a period of time by the collaborators.

Note that this approach can even be used by a group working in the same place, and so it could also be placed in the same-place, different-time quadrant of the Applegate matrix.

6.4. Conclusions on Web-based Visualization

The Web has become an important environment for distributed computing in particular, and distributed visualization in particular. We have seen how systems can map on to the client-server architecture in a variety of ways - with a major distinction between Client-based and Server-based systems.

The Web is fundamentally an environment for cooperation. We have seen that it can be used for distributed cooperative visualization in a variety of ways: the synchronous collaboration afforded by MVEs can be invoked in a Web environment; and asynchronous collaboration can be achieved in Server-based systems, using a stored tree of explorations. Recently Lovegrove and Brodlie[44] have studied how multi-user VRML can be exploited to provide synchronous collaboration in a Server-based system.

These are all very explicit forms of collaboration, where there is cooperation in the excution of the visualization by the interested parties. However the Web also allows some less obvious collaboration. For example, simply making available a dataset at a URL makes it available at any time, any place to collaborators. The Vis-a-Web system of Pagendarm allows the merging of visualizations created by different software at different places: this is almost cooperation between providers of visualization software.

7. Collaborative Visualization and Virtual Reality

For many visualization problems, VR is a promising technology. Gallop[26] provides an introduction to the application of VR in this field. The CAVE[18], mentioned earlier, is extensively used for scientific visualization. The need to cooperate at a distance is no less when VR is involved.

VRML on the Web provides a way to cooperate using 3D visualization with the participants at different places, and working at different times and has been discussed in section 6. This approach has the advantage of relying on a formal (ISO/IEC) and accepted standard.

Several projects have experimented with cooperative VR. This allows people at different locations to set up representations of themselves (avatars), together explore the same world and communicate with each other about what they observe. For example DIVE[59, 14, 15] allows several networked participants to interact in a virtual world over the Internet. A number of approaches to multi-user VRML worlds have also been described, not specifically with visualization applications in mind[22]. Broll, for example, has worked extensively in this area and has developed a multi-user VRML browser which provides support for the representation of users by avatars and communication between users by integrated audio and text chat facilities. The system uses a distributed worlds transfer protocol which supports the reliable transmission of large data files between peers, the reliable or unreliable transfer of messages between large numbers of participants and the transfer of stream data such as audio[10, 11].

This kind of technology can be applied to visualization results if the geometric output can be input to the cooperative VR system. The multiple users in the virtual world would together explore the abstract scene created by the visualization system.

However this is a passive way of approaching visualization. The work by COVISA, by COVISE, by MANICORAL and at SDSC (see section 4) shows that users want shared control over the process not just shared exploration of the output. The experience of the VIVRE project[67] also shows that users studying large problems with VR expect to use the virtual environment to exercise control over the visualization process. It is therefore a natural step to allow dynamic cooperative control of the visualization process from the virtual environment. However this is not common as yet. The University of Stuttgart are extending COVISE (discussed in section 4) by adding VR capability to the system [56, 57]. The user works in a CAVE and some of the user controls in COVISE are available in the virtual environment. The user can access further controls outside the CAVE.

The Studierstube project[65] allows multiple users in a "study room" to cooperate studying 3-dimensional scientific visualization. Each participant wears an individually head-tracked see-through head mounted display and thus has their own personal view. It is plain that each person's view is individually rendered thus increasing the computational load. New visualization calculations can be triggered from any participant's virtual environment.

8. Distributed Collaboratories

Distributed Cooperative Visualization is part of a wider vision for the use of computer supported cooperative working tools in science and engineering. In this section we give a glimpse of that wider picture, illustrated by a specific programme in the USA.

A report[16] written in 1993 by the National Research Council noted:

> The fusion of computers and electronic communications has the potential to dramatically enhance the output and productivity of U.S. researchers. A major step toward realizing that potential can come from combining the interests of the scientific community at large with those of the computer science and engineering community to create integrated, tool-oriented computing and communication systems to support scientific collaboration. Such systems can be called "collaboratories".

The motivation behind this view is the fact that a significant proportion of modern science involves large and expensive instruments, often available only at a single location, and teams of scientists drawn from universities, national laboratories and industry. Geographical separation can be a serious impediment to collaboration. Face to face meetings are expensive in terms of travel costs, time and are difficult to schedule. The vision of distributed collaboratories is to provide a widely distributed environment in which people, instrumentation, and information can flow and interact as easily as they can when all the critical resources are co-located. This vision led to the creation of the Distributed Collaboratory Experiment Environments (DCEE) programme by the US Department of Energy[37]. This programme has now terminated and the vision is being carried forward in the DOE2000 programme.

The DCEE programme consisted of two threads: testbeds which helped to introduce and integrate technology needed for remote operation of facilities, collaboration and information sharing and technology projects addressing specific functionality and components not previously available. The testbed projects included medium scale environments (tens of scientists and students) in the PNL Environmental and Molecular Sciences Laboratory, the ANL Advanced Analytical Electron Microscope and the LBL Advanced Light Source, together with a large scale experiment (a team of hundreds) in the General Atomics D-IIID Tokamak Fusion Facility. Technologies identified included: distributed, cross-platform electronic notebooks to collect, catalogue and share results and insights, multi-party data communications protocols, open and secure global file systems, and use of shared space and VR approaches to improve the sense of awareness and engagement in interactions.

One of the key foci in the DCEE programme, which is carried into the DOE2000 programme, was the Electronic Notebook. The aim here is to build an electronic replacement for the traditional paper-based laboratory notebook into which scientists would paste experimental results and describe in detail their work in a verifiable timeline. The laboratory notebook is often the only piece of evidence that can be offered in court cases involving patent law. The interpretation of such requirements in the past has made sharing of notebooks difficult.

There is a clear need and desire to integrate visualization into such environments. The DOE2000 White Paper talks about mechanisms to support collaborative work including reliable telepresence software and support for immersive visualization applications. Notebooks being developed in DOE2000 projects seem to be based on Corba object technology, Java and Web technology. Integration is a significant challenge in this area: integration at the protocol, object management, data base and user interface levels.

9. Conclusion

As networks become more pervasive and collaboration becomes more and more important in many areas of life, it is natural to ask what computing can contribute. In many areas of science and engineering and increasingly in other areas too, visualization plays a key mediating role in communications between humans.

As the area of distributed cooperative visualization matures, we can expect the research ideas discussed in this report to translate into products in the market place. Indeed the COVISA extension of IRIS Explorer is already available for UNIX implementations at the University of Leeds IRIS Explorer Centre of Excellence [62], and the next release of IRIS Explorer is expected to include COVISA for both NT and

Unix. The usage of these systems in the field is likely to stimulate a further round of research, as strengths and weaknesses emerge.

We have attempted in this STAR to give a review of approaches to supporting the use of visualization in collaborative situations, through what we have termed distributed cooperative visualization. We have examined approaches based on MVEs and web-based approaches. This is a field in which we may expect to see continued growth over the coming years, as the web develops and (hopefully) high quality networking becomes more available and more affordable.

It is interesting to note that CSCW technology was used in the preparation of this STAR. The schedules of the authors were such that it was impossible to find a date on which to meet face-to-face. We therefore had to resort to a combination of audio-video conference sessions (conducted in small conference suites at the University of Leeds and RAL over ISDN connections) and email for the exchange of draft sections of the document. The reader is left to judge the efficacy of this approach from the point of view of the end results! For the authors, this was an effective use of time, though it was very helpful that the authors knew each other well and have built up a common understanding of at least some of the material over a period of years.

Acknowledgements

DAD and JRG gratefully acknowledge fruitful discussions with their colleagues at RAL involved in the MANICORAL project (EU Telematics Programme RE1006) and with other partners in the project, which have contributed greatly to shaping their understanding of this area. KWB and JDW likewise thank all those who have contributed to our understanding of the subject - notably Helen Wright (now of University of Hull), Jeremy Walton and colleagues at NAG Ltd, and many postgraduate and undergraduate students at the University of Leeds. In particular the COVISA research project, funded by EPSRC under the ROPA initiative and carried out by the two of us together with Helen Wright, gave us the means to study collaborative visualization in depth.

References

1. G. Abram and L. Treinish. An Extended Dataflow Architecture for Data Analysis and Visualization. *Computer Graphics*, 29(2):17 – 21, 1995.
2. M. Agrawala, A.C. Beers, B. Frohlich, P. Hanrahan, I. McDowall, and M. Bolas. The Two-User Responsive Workbench: Support for Collaboration Through Independent Views of a Shared Space. In *Computer Graphics Proceedings, Annual Conference Series, 1997*. ACM Press, 1997.
3. Cheong S. Ang, David C. Martin, and Michael D. Doyle. Integrated control of distributed volume visualization through the world wide web. In R. Daniel Bergeron and Arie E. Kaufman, editors, *Proceedings of IEEE Visualization 94*, 1994.
4. L.M. Applegate. Technology Support for Cooperative Work: A Framework for Studying Introduction and Assimilation in Organizations. *Journal Organizational Computing*, 1:11–39, 1991.
5. G. Bancroft, F. Merrit, T. Plessel, P. Kelaita, R. McCabe, and A. Globus. FAST: A Multi-Processing Environment for Visualization of CFD. In *Proceedings of Visualization 90*. IEEE Press, 1990.
6. D. Bergeron. Visualization reference models (panel session position statement). In G.M. Nielson and D. Bergeron, editors, *Proceedings of Visualization '93*. IEEE Computer Science Press, 1993.
7. S.A Bly, S.R. Harrison, and S. Irwin. MediaSpaces: Bringing People Together in a Video, Audio and Computing Environment. *Communications of the ACM*, 36(1):28 – 47, 1993.
8. K.W. Brodlie. Visualization over the World Wide Web. In *Proceedings of the Dagstuhl Workshop 1997, to appear*, 1998.
9. K.W. Brodlie, D.L. Fisher, G.G. Tolton, and T.W. Lambert. The development of the NAG Graphical Supplement. *Computer Graphics Forum*, 1(3):133–142, 1982.
10. W. Broll. Multi-user Virtual Reality on the Internet. In *ERCIM News*, number 31, page 30, 1997.
11. W. Broll. DWTP - An Internet Protocol for Shared Virtual Environments. In *Proceedings of the Virtual Reality Modelling Language Symposium 1998 (VRML '98)*, 1998.
12. J. Buckett, I.L.C. Campbell, T.J. Watson, M.A. Sasse, V. Hardman, and A. Watson. ReLaTe: Remote Language Teaching over SuperJANET. In *Proceedings of UKERNA Networkshop 23*, 1995.
13. G. Cameron. Modular Visualization Environments: Past, Present, and Future. *Computer Graphics*, 29(2), 1995.
14. C. Carlsson and Hagsand O. DIVE - A Multi User Virtual Reality System. In *Proceedings of VRAIS '93*. IEEE Computer Science Press, September 1993.
15. C. Carlsson and Hagsand O. DIVE - A Platform for Multi-User Virtual Environments. *Computers and Graphics*, 17(6):663–669, 1993.
16. V.G. Cerf et al. *National Collaboratories - Applying Information Technology for Scientific Research*. National Research Council, National Academy Press, Washington, D.C., 1993.

17. B. M. Collins. Data visualisation - has it all been seen before? In *Animation and Scientific Visualisation*. British Computer Society, State of the Art Report, 1992.

18. C. Cruz-Neira, D.J. Sandin, and T.A. DeFanti. Surrond-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 135 – 142. ACM Press, 1993.

19. D.A. Duce, J.R. Gallop, I.J. Johnson, K. Robinson, C.D. Seelig, and C.S. Cooper. Distributed Cooperative Visualization - The MANICORAL Approach. In *Eurographics UK Chapter Conference 1998*. University of Leeds, 1998.

20. D.A. Duce, J.R. Gallop, I.J. Johnson, K. Robinson, C.D. Seelig, and C.S. Cooper. Distributed Cooperative Visualization - Experiences and Issues from MANICORAL Project. In *Proceedings of the Eurographics Workshop on Visualization in Scientific Computing*. Eurographics Association, 1998.

21. D.A. Duce, D. Giorgetti, C.S. Cooper, J.R. Gallop, I.J. Johnson, K. Robinson, and Seelig. Reference Models for distributed Cooperative Visualization. Computer Graphics Forum, in press, 1998.

22. D.A. Duce, K. Kansy, and M.D. Wilson. Report and Recommendations of the VRML Workshop 29/30 January 1997. Technical Report 02/97-R049, ERCIM, B.P. 105, F-78153, Le Chesnay Cedex, France, 1997. Available via the web at http://www-ercim.inria.fr.

23. D.D.E. Long et al. REINAS the real-time environmental information network and analysis system. In *Proceedings of COMPCON*, pages 482 – 487. IEEE, 1995.

24. D. Foulser. IRIS Explorer: A Framework for Investigation. *Computer Graphics*, 29(2):13 – 16, 1995.

25. H. Fuchs. Building Telepresence Systems: Translating Science Fiction Ideas into Reality. *Computer Graphics Forum*, 16(3):C–189, 1997.

26. J. Gallop. *Virtual Reality - its Application to Scientific Visualization*. Eurographics Tutorial Notes, EG96 TN3. Eurographics Association, 1996. ISSN 1017-4656.

27. M.J. Gerald-Yamasaki. Cooperative visualization of computational fluid dynamics. *Computer Graphics Forum*, 12(3):C–497 – C–508, 1993.

28. M. Grave. Shared Data Spaces for Distributed Computing and Parallelism in Scientific Visualization Systems. *CWI Quarterly*, 7(2):175 – 185, 1994.

29. S. Greenberg, S. Hayne, and R. Rada. *Groupware for Real-time Drawing: A Designer's Guide*. McGraw-Hill, 1995.

30. R.B. Haber and D.A. McNabb. Visualization idioms: A conceptual model for scientific visualization systems. In B.Shriver G.M.Nielson and L.Rosenblum, editors, *Visualization in Scientific Computing*, pages 74 – 93. IEEE Computer Society Press, Los Alamitos, CA, 1990.

31. V. Hardman, M.A. Sasse, and I. Kouvelas. Successful Multiparty Audio Communication over the Internet. *Communications of the ACM*, 41(5):74 – 80, 1998.

32. P.L. Isenhour, J. Begole, W.S. Heagy, and C.A. Shaffer. Sieve: A java-based collaborative visualization environment. In *Late Breaking Hot Topics Proceedings, IEEE Visualization 97*, 1997.

33. S. Jacobs, M. Gebhardt, S. Kethers, and W. Rzasa. Filling HTML forms simultaneously: CoWeb - architecture and functionality. *Computer Networks and ISDN Systems*, 28(7-11):1385–1395, 1996.

34. M. Jern. Information Drill-down using Web tools. In *From Desktop to Webtop: Virtual Environments on the Internet, WWW and Networks, BCS Computer Graphics and Displays Group International Conference*, 1997.

35. M. Jern. 3D Data Visualization on the Web. In *Multimedia Modelling '98, Lausanne, Switzerland, October 1998*, 1998. Proceedings to be published by IEEE.

36. Abraham Kee. Visualization over WWW using Java. Msc thesis, School of Computer Studies, University of Leeds, 1996.

37. R.T. Kouzes, J.D. Myers, and W.A. Wulf. Collaboratories: Doing science on the internet. *IEEE Computer*, 29(8):40 – 46, August 1996.

38. W. Krueger, C.-A. Bohn, B. Froehlich, H. Schueth, W. Strass, and G. Wesche. The Responsive Workbench: A Virtual Work Environment. *IEEE Computer*, 28(7):42 – 48, 1995.

39. U. Lang et al. Perspectives of collaborative supercomputing and networking in European Aerospace research and industry. *Future Generation Computer Systems*, 11:419 – 430, 1995.

40. W. Lefer. A Distributed Architecture for a Web-based Visualization Service. In *Proceedings of the Eurographics Workshop on Visualization in Scientific Computing*. Eurographics Association, 1998.

41. V.D. Lehner and T.A. DeFanti. Distributed Virtual Reality: Supporting Remote Collaboration in Vehicle Design. *IEEE Computer Graphics and Applications*, 17(2):13 – 17, 1997.

42. T. Liao. WebCanal: a Multicast Web Application. In *Proceedings of the Sixth International World Wide Web Conference*, 1997.

43. H.D. Lord. Improving the Application Development Process with Modular Visualization Environments. *Computer Graphics*, 29(2):10 – 12, 1995.

44. S. Lovegrove and K.W. Brodlie. Collaborative research within a sustainable community: Interactive multi-user VRML and visualization. In *Proceedings of Eurographics UK Chapter 1998 Conference*, pages 53–68, 1998.

45. B. H. McCormick, T. A. DeFanti, and M. D. Brown. Visualization in scientific computing. *Computer Graphics*, 21(6), 1987.

46. C.K. Michaels and M.J. Bailey. VizWiz: A java applet for interactive 3d scientific visualization over the web. In *Proceedings of IEEE Visualization 97*, pages 261–268, 1997.

47. H.-G. Pagendarm. Visualization within environments supporting human communication. *To be published in Future Generation Computer Systems, Amsterdam*, 1998.

48. H.-G. Pagendarm and F.H. Post. Comparative Visualization - Approaches and Examples. In M. Göbel, Müller H., and B. Urban, editors, *Visualization in Scientific Computing*. Springer-Verlag, Wien, 1995.

49. A. Pang. Spray rendering. *IEEE Computer Graphics and Applications*, 14(5):57 – 63, 1994.

50. A.T. Pang and C.M. Wittenbrink. Collaborative 3D visualization with CSpray. *IEEE Computer Graphics and Applications*, 17(2):32 – 41, 1997.

51. P. Parnes, M. Mattsson, K. Synnes, and D. Schefstrom. The mWeb Presentation Framework. In *Proceedings of the Sixth International World Wide Web Conference*, 1997.

52. B. E. Rogowitz. The psychology of visualization (panel session position statement). In G. M Nielson and D. Bergeron, editors, *Proceedings of Visualization '93*. IEEE Computer Science Press, 1993.

53. A. Scott and H. Wolf. Collaborative Browsing in the World Wide Web. In *Proceedings of the 8th Joint European Networking Conference*, 1997.

54. Will Shroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice-Hall, 1996.

55. P. Stanton. Java visualization software. Final year project, School of Computer Studies, University of Leeds, 1997.

56. See the COVER project's website http://www.hlrs.de/structure/organisation/vis/covise/features/cover/.

57. See the COVISE project's website http: //www.hlrs.de/structure/organisation/vis/covise/.

58. See the CurVis Web Site http://www.informatik.uni-rostock.de/Projekte/movi/IIS/curvisrdr.html.

59. See the DIVE website http://www.sics.se/dce/dive/online/diveinfo.html.

60. See the FAST project's website http://science.nas.nasa.gov/Software/FAST/FASTexpeditions/. The information about Remote Collaboration is found by following a link to Frequently Asked Questions.

61. See the IBM Data Explorer Weather Forecasting Web Site http://www.research.ibm.com/weather/vis/w_vis.htm.

62. See the IRIS Explorer Centre of Excellence Web Site http://www.scs.leeds.ac.uk/iecoe.

63. See the SciVis Web Site http://kopernik.npac.syr.edu:8888/scivis/.

64. See the SDSC project's website http://www.sdsc.edu/PET/scivis/page01.html.

65. See the Studierstube website http://www.cg.tuwien.ac.at/research/vr/studierstube/AVS/html/.

66. See the Vis-5D Web Site http://www.ssec.wisc.edu/ billh/view5d.html.

67. See the VIVRE website http://www.tessella.co.uk/projects/vivre/index.htm.

68. See the VTK Web Site http://www.kitware.com/vtk.html.

69. See the website http://media.it.kth.se/SONAH/ANALYSYS/race/pl7/present/pagein/images.htm.

70. J. Trapp and H.-G. Pagendarm. A prototype for a WWW-based visualization service. In W.Lefer and M. Grave, editors, *Visualization in Scientific Computing '97*, pages 21–30. Springer, Wien, 1997.

71. C. Upson et al. The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9(4):30 – 42, July 1989.

72. J. Walton and D. Knight. Rock 'n' roll: Using VRML2.0 for visualization. IRIS Explorer Technical Report TR27 IETR (NP3159), NAG Ltd, 1997.

73. R. Weggenkittl and E. Groeller. Fast oriented line integral convolution for vector field visualization via the internet. In *Proceedings of IEEE Visualization 97*, pages 309–316. ACM Press, 1997.

74. A. Wierse. Performance of the COVISE visualization system under different conditions. In G. Grinstein

and R.F. Erbacher, editors, *Visual Data Exploration and Analysis II*, volume SPIE 2410, pages 218 – 229, 1995.

75. A. Wierse, U. Lang, and Rühle. A System Architecture for Data-oriented Visualization. In J.P. Lee and G.G. Grinstein, editors, *Database Issues for Data Visualization, IEEE Visualization '93 Workshop*, volume 871 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

76. J. Wood, H. Wright, and K. Brodlie. CSCV - Computer Supported Cooperative Visualization. In *BCS Displays Group Conference on Visualization and Modelling*, 1995.

77. J. Wood, H. Wright, and K. Brodlie. Improving Visualization Through Collaboration. In *Eurographics Workshop on Scientific Visualization*, 1995.

78. Jason Wood, Ken Brodlie, and Helen Wright. Visualization over the World Wide Web and its application to environmental data. In R.Yagel and G.M. Nielson, editors, *Proceedings of IEEE Visualization 1996 Conference*, pages 81–86, San Francisco, CA, 1996. ACM Press.

79. Jason Wood, Helen Wright, and Ken Brodlie. Collaborative visualization. In *Proceedings of IEEE Visualization 97*, pages 253–259. ACM Press, 1997.

80. J.D. Wood. *Collaborative Visualization*. PhD thesis, School of Computer Studies, University of Leeds, U.K., 1998.

81. Alan Yeo. Client-based web visualization. Msc thesis, School of Computer Studies, University of Leeds, 1998.

82. M. Young, D. Argiro, and J. Worley. An Object Oriented Visual Programming Language Toolkit. *Computer Graphics*, 29(2):25–28, 1995.