Collaboration as a Second Thought

Mark Hereld, Michael E. Papka, Thomas D. Uram
Mathematics and Computer Science Division
Argonne National Laboratory
{hereld, papka, turam}@mcs.anl.gov

ABSTRACT

Collaboration is often an afterthought to a project or development. In this paper we describe and analyze our experiences in developing collaborative technologies, most often involving the sharing of visual information. We have often developed these in a context that required us to retrofit existing analysis applications with collaboration capabilities. This approach, though fruitful, is time-consuming, expensive, and often difficult to re-apply elsewhere – it is just hard to change an existing application. One way to make such an effort easier is to package the collaborative components as a kit that can be leveraged on a case-by-case basis. The fixed interface provided by a well-designed toolkit eases the integration process by providing an unchanging and familiar set of components to deploy. Better still, we find, are approaches that require no modification of applications while providing rich and powerful means for sharing information with collaborators. We discuss three separate and illuminating examples that meet this criterion in different ways: (1) building the collaborative potential into the underlying abstractions on which operating systems are built, (2) building tools that live side-by-side with any application in the context provided by the operating system, or by (3) building information tools that use collaborative modalities to better integrate with our workflow. These are probably not the only options, but they all derive from an approach where collaboration is considered early in the design process and therefore manifests itself deep in the computing infrastructure giving it a wider cast.

KEYWORDS: Collaboration, Visualization, Best-Practices, Human-Computer Interaction

1. INTRODUCTION

The 2007 report on *Visualization and Knowledge Discovery* [1] concludes that basic research is needed in several areas to enable continued scientific discovery, given the current trajectories of data produced by simulations and collected from sensors. Interaction and collaboration are at the top of the list:

> *A new generation of visualization and data exploration tools are needed to significantly enhance interaction and collaboration between these distributed scientists, their data, and their computational environments.*

What is interesting is that this statement clearly acknowledges three components of a successful collaboration: scientists, data, and the environment itself. Building on a long history of creating collaborative environments to support science, we discuss in this paper the challenges associated with taking an existing tool and making it "collaborative." We present the various methods we have used, from leveraging tools that make an application appear to be collaborative, to adding infrastructure in an ad hoc manner to enable collaborative use, to building on a toolkit or framework that supports collaboration. We start with a brief background on the collaborative spaces we have explored. We then relate our experiences in retrofitting tools to make them collaborative. We discuss a few environments and tools that have collaboration at their very core. From our experiences we suggest the next steps in supporting effective collaboration.

2. BACKGROUND

Starting in 1994 we began to experiment with the construction of collaborative virtual environments. Our early work progressed from the first CAVE-to-CAVE application, to the development of the CAVEcomm library [2] and the advanced ManyWorlds framework [3], and finally to a desktop system called Metro. At the same time we were looking at how to leverage efforts in the community in the construction of middleware to support distributed workspaces [4].

These experiences made us realize that, while considerable commercial effort was being devoted to

solving the desktop-to-desktop scenario, certain areas were still being ignored: (1) group-to-group collaborations among people at different institutions and (2) a way to integrate tools people use when meeting face to face. Video conferencing solutions with single cameras, click to talk, and a single microphone did not meet such needs. They were also not very engaging or natural environments.

To address this situation, we initiated the Access Grid project [5]. Our objective was to design and build a deployable tool to enable groups of individuals at remote locations to collaborate and work as if they were collocated. The Access Grid is an open-source research project dedicated to enabling collaboration between groups of individuals, through the sharing of audio, video, text, and applications. The Access Grid represents a scalable solution for collaboration ranging from the desktop to room-size suites. The development of the Access Grid toolkit, which is now in version 3.1, represents significant experience in working with a large community of diverse users. The Access Grid currently supports over 2,000 users in over 50 different countries. Initial prototypes of integration of visualization infrastructure with the framework have been demonstrated over the past five years [6,7] and more recently in a production environment for the volumetric rendering of medical images [8]. In conjunction with this effort we have been planning and prototyping community-accessible visualization solutions to simplify access to resources [9]. To date, collaboration technology has not advanced beyond the situation that was the impetus for the development of the Access Grid.

3. COLLABORATION RETROFITTING

Collaboration has long been an important element of scientific research; generally this has been a local collaboration in which collocated colleagues share scientific and computational resources. As collaborations become more distributed, it is essential that the capability continue. In many cases, however, the applications that scientists rely on are not collaborative and often are not even network-aware. These applications must be retrofitted for collaboration so that people in geographically distributed locations can use them and have a common sense of the workflow.

3.1. Manual Retrofitting

Multiple approaches to manual retrofitting have been pursued. Two common approaches are the use of remote screen buffers and implementation of collaborative facilities within the application. The most trivial is that of screen sharing à la virtual network computing (VNC), in which the content of the display is captured and transmitted to remote participants, from whom interactions can be captured and transmitted back to the source application. The wide applicability of this solution is appealing, as it is largely application- and platform-independent. It does not, however, preserve any shared representation of the application state, essential to the continuing pursuits of the collaborative endeavor. VNC also faces well-known security and connectivity challenges.

A second approach for equipping an application for collaboration is to implement this functionality directly in the application code. This approach has its merits, primary tight integration and full control of the application. We have used this approach in numerous instances, for example in writing stream-processing units for Chromium to stream-rendered frames as video. The downside of this approach is its labor-intensive nature, in that it requires not only design of novel collaborative facilities but also deep knowledge of the target application code. Having followed this approach in several instances, we have identified common facilities and patterns, clarifying the need for a collaboration toolkit.

3.2. Retrofitting with a Collaboration Toolkit

The Access Grid establishes a collaborative environment with audio, video, text chat, shared data, and shared applications. Beyond the typical user tools for collaboration, the Access Grid includes a toolkit for developing shared applications. The collaborative components of the toolkit are key to establishing and maintaining shared distributed state: a centralized store of shared application state and an event distribution service. The centralized application state consists of a memory-resident keyword-value store. The storage interface itself places minimal constraints on the structure, content, and format of the data, instead leaving these decisions to the application.

The event service is similar in its treatment of data: it includes facilities for addressing and messaging, but regards the data payload as entirely opaque. Performance was a key consideration in the design of the event service; using lightweight data formats, the event service is able to deliver messages between users distributed around the planet with minimal overhead. A further benefit of the event service is that it avoids firewall problems by relying strictly on outgoing connections from users. These facilities provide the underpinnings necessary for applications to be extended for collaboration.

Implementing collaborative facilities in an existing application is significantly simplified when one need not be concerned with the format of data or means of distribution. In the next section, we give examples of

standalone applications that have been extended with collaborative facilities using the application development components of the Access Grid.

4. EXPERIENCE

Collaboration can be divided into two functionally distinct layers: remote presentation and remote interaction. Remote presentation involves simply sharing the content of one's work with remote viewers, presumably with some out-of-band mechanism for discussing the content. This "half-duplex" interaction is often an acceptable level of collaboration in situations involving a single expert or localized data or application base and a group of remote "observers."

Remote interaction involves bidirectional or "full duplex" interaction, in which the remote collaborators can fully control the application being shared. This scenario is an extension of remote presentation in that when all collaborating participants have equal expertise with the application or data being presented, any one of them can take over and discuss the shared content.

The following sections describe our experience with each of these mechanisms.

4.1. Streaming ParaView-rendered Data

ParaView is a widely used application for visualization of scientific datasets (Fig. 1). These datasets are often large and reside on remote data storage resources, with which the ParaView client interacts through a built-in client-server mechanism, such that the server reads the data and renders it and the client simply displays it.
Scientific visualization is typically undertaken by visualization experts operating on data that belongs to scientists. These two groups work together on the development of the visualization to ensure that it accurately represents the data, usually over a period of time proportionate to the size and complexity of the dataset.

As research groups become increasingly distributed, these interactions must also become distributed. To model these interactions in a distributed environment, the visualization expert and scientist require a mechanism for reviewing intermediate visualization results and adjusting the space, time, and color representations from their home institutions.

The first step in our solution is to allow the visualization expert to work in the usual fashion, in this case using ParaView, with an extension that streams the resulting visualization over the network as video to the remote scientists. The scientists run an application that can receive and display the incoming visualization data. The visualization frames are streamed by using a video codec that optimizes for bandwidth consumption by limiting the data it sends according to changes between frames (e.g., motion coding). This allows the stream to achieve a high frame rate and low latency, making it a convenient solution for collaborative visualization (Fig. 2).

Used in conjunction with the Access Grid, which includes the application that receives and displays the incoming visualization video, the visualization expert and the scientist(s) can interact with each other and the visualization in a fashion similar to being in the same physical room.

Subsequent development will focus on enabling the remote scientists to take control over ParaView and adjust the visualization, as over time the domain scientists will become more comfortable with the visualization tools and it will be more natural and faster for them to interact with the visualization themselves.

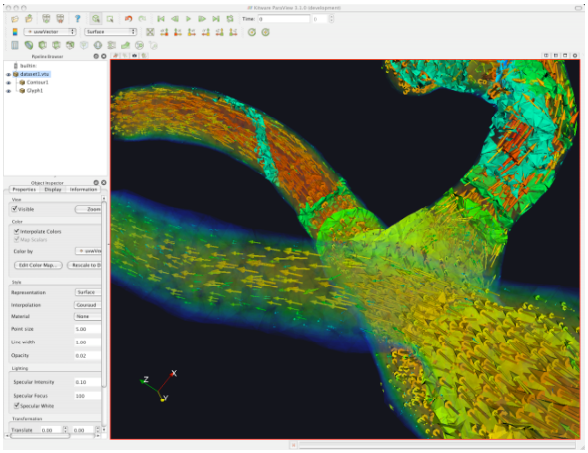

Figure 1. Screen capture of ParaView application being used to look at data from a simulation of blood flow within an artery. (Data provided by George Karniadakis - Brown University)

4.2. Access Grid Shared Applications

Here we give four examples of shared applications that have been built with the collaborative components provided by the Access Grid: Shared Presentation, Shared Rasmol, Shared Gnuplot, and vl3.

Shared Presentation is a tool for distributed control of slide presentations. This application uses the data store for state, such as the location of the set of slides (e.g., a URL), the current slide number, and the identity of the presenter (for floor control). The event service is used to

Figure 2. Photos of users participating in collaborative ParaView session; the user sitting at his desk is sharing the visualization results with two colleagues at a different location.

communicate events such as the loading of a slide set, advancing to the next slide, and floor control changes.

Rasmol is an application used by biologists for three-dimensional visualization of molecular structure. We have built Shared Rasmol to allow biologists to collaboratively view molecular models from remote locations. Shared Rasmol opens model files from the Access Grid Venue datastore and tracks application interactions for rotating, panning, and zooming the model and changing the display of the model. These interactions are communicated to remote users where they are pumped into the receiving Rasmol instance to effect the same view. The interactions are communicated over the event service, which must bear the heavy load of frequent updates to the model position and transformation resulting from mouse interactions.

Gnuplot has a long history in graphical viewing of scientific data by scientists in many domains. Shared Gnuplot accepts input from any member of a group of users and displays the resulting plot at all remote sites. As with other shared applications, this allows one user with expert knowledge in either the tool (Gnuplot) or the domain data to share his expertise with the group.

The vl3 tool is a distributed, cluster-based volume visualization application with a collaborative frontend. The rendering engine is run on the cluster, and the user interface communicates changes in the view back to it for rendering updates. Discovery of the vl3 "session" occurs through the Access Grid Venue (as is typical of AG shared applications). When a user joins the vl3 session, the client launches against addresses stored in the shared application state, and interactions with the client propagate to remote collaborators over the Access Grid event service.

4.3. Results

The examples above describe two approaches to building collaborative applications: remote presentation and remote interaction. A hybrid of these two approaches, best exemplified in the case of vl3, may be the best approach going forward. Streaming the relevant application interface to remote participants as video optimizes for presentation and bandwidth but precludes interaction. The Access Grid facilities used by the shared applications described above arose out of many instances of shared application development and provide the necessary support for remote interaction.

5. FUTURE POSSIBILITIES

Interesting environments can arise when one begins by considering collaboration instead of ending there. In this section we present three ideas for collaborative technologies. We think that they create collaborative experiences that are very different from what we have seen, even though they rely heavily on ideas that have been tried in part and in other forms. The three examples below illustrate powerful collaborative capabilities available to all applications in conventional personal computing environments.

5.1. The Vast Pixel Savannah

How can we share what we are seeing on our display screens in a simple and fluid way? This question lies at the heart of collaborative visualization.

The need for a new paradigm in handling pixels is being expressed in several areas of ongoing work. At the more traditional end of the scale are the solutions that consider end-point displays as containing graphics horsepower according to the usual workstation standards. Multi-display systems using WireGL, Chromium, DMG, and

similar solutions are of this ilk. A central problem for architectures based on this paradigm is that operating system integration with the display adaptor has evolved to provide powerful graphics pipelines from application to pixels that are difficult to break into. At the other end of the scale, solutions that put pixels in the prime position (as opposed to graphics primitives) are providing perhaps the most flexible and neutral paradigm, typically at the expense of performance. SAGE, VNC, and the many progeny of VNC are among the approaches that fit into this category. The venerable and ancient X lies between these extremes.

With the vast pixel savannah (VPS), we consider a model where pixels are primary—shipped à la carte across the network—and introduce the notion of a global address space for pixels that is accessible to all display devices. Shown in Figure 3 is an example of two conventional work planes, a laptop and a dual headed workstation, that are making use of two windows onto the VPS. Each is windowed onto more or less permanent *homestead* plots, which in this case are temporarily connected via a bridge (labeled *wormhole*) to facilitate passing windows. Each is additionally windowed onto a larger shared pixel plot (labeled *playground*) that might be the target of a large visualization that these two collaborators are exploring. Thrown in for the purpose of illustration are plots in the VPS corresponding to the display of a cell phone and a public digital billboard
. Pixel real estate might be apportioned by using DNS-like services over the Internet. Every display device that adopts this model would face many of the same administrative questions that it faces with respect to network address: where to begin, if and when to change the position of its view onto the VPS, who to ask if it doesn't know what to do, and what coordinates are legal.

Taking this view of display space would enable many useful possibilities. One's display becomes a window onto this terrain. One may own a private plot, or pixel homestead. Bridges between regions create expedient new relationships by manipulating the topology of the VPS. An extreme interpretation of the VPS that may open its use to new and interesting possibilities is as a kind of collaborative terrain along the lines of Second Life.

5.2. Pixel Porting

Sometimes collaboration is best served by sharing only a small piece of a visual representation right now and without any prior warning. The barrier to such impromptu visual collaborations is too high for most people's taste, and so it rarely contributes to the collaborative workflow.

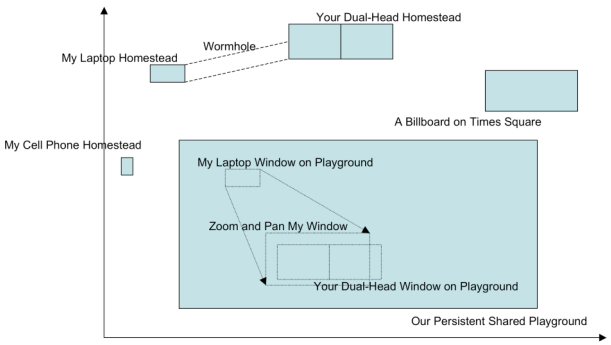

Figure 3. The Vast Pixel Savannah, a universally shared persistent visual space.

Standing between collaborators and a fluid flow of shared visual experience are a host of issues including non-uniform interfaces to applications, computing platform differences, the semantics of the desktop metaphor, and the mechanics of keyboard and mouse. Issues of personal preference also play a key role in this problem. One can see instances in existing applications and modes of human-computer-human interaction that capture useful styles. Instant messaging has many properties that could aid us here, the spontaneity afforded by an open channel and its lightweight interface among them. Keyboard shortcut tools (Quicksilver comes instantly to mind) can be extremely powerful; though can present a bewildering if not daunting array of magical keystrokes that is off-putting to many. Direct selection of pixels from the user's field of view is an exceptionally useful tool because it has the power to represent exactly what interests while skipping over the encumbrances of the desktop metaphor – focus is on the flat visual imagery.

We believe that a useful tool capable of promoting frictionless sharing of visual data can be constructed out of the best aspects of these many components. Here is a sketch of what we envision, in the form of a use case.

You see something interesting on your screen that you would like to share with a collaborator. You hold the Ctrl-Option meta-keys while dragging a selection box over the area of your screen that contains the important content. The highlighted region (a pixel portal) can then be dragged onto your IM client where it resolves to an existing conversation or creates a new one as circumstances dictate. In Figure 4, the provider on the right (*you*) can resize or move using corner handles. The widget bar below the portal window contains selectors for still, new snap, continuous, and remote control. The widget bar below the destination port (*mine*), if enabled by the provider, allows the subscriber to steer the pixel portal's mouth end around the window or screen on the provider's laptop. Decorating the conversation now is an active region with the contents of the selected pixel

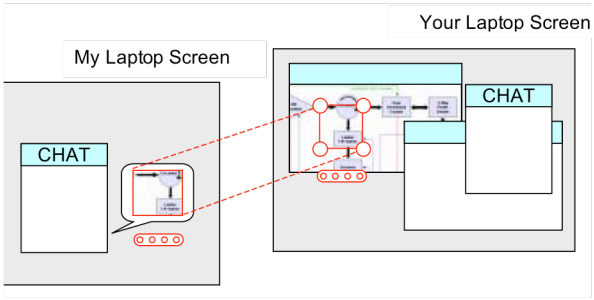

Figure 4. Sketch of a pixel port connection.

portal; it can be a snapshot of the contents at the moment of selection or, at your option, a constantly updated image mirroring your selection. At any time you can resize or move the pixel portal with instantaneous effect on what your collaborator sees. You may allow your collaborator to steer. Pixel portals can be managed by a mechanism parallel to the ubiquitous clipboard, or they can be redirected to other or additional collaborators.
With this notion in place, many potentially useful embellishments are possible.

5.3. IM Everywhere

As our research workflows become increasingly complex and intertwined, not only with our human collaborators, but also with the range of computing processes that facilitate all aspects of our daily chores, there is increasing need for human-centered approaches in order to relieve the mental overhead that attends this complexity.

Perhaps it is opportune to start thinking about broadening the sphere of influence of collaboration research to include the non-human partners in our work. Human-centered approaches to computing are increasingly feasible as technologies in natural languages, agents, security, ubiquitous computing, and grid-computing (to name a few) are advanced and integrated into our common computing fabric.

The idea embodied in IM Everywhere is to deploy our agents behind the natural interface provided by the IM client. By packaging our computing aids as agents and wrapping them in the conversational interface we use for our human collaborations we reduce the number of different command and control modes that we are required to master, reduce the number and depth of context switches required to manage our multi-tasking, and begin an evolutionary process that will ultimately enable us to focus more exclusively on our research problems and less on the problems created by the technologies we hurl at our research problems.

6. CONCLUSIONS

Current collaborative infrastructure available today is not enough; we need collaborative services that integrate with scientific workflow and data management systems. Systems must support and promote remote and collaborative visualization and have algorithm and infrastructure optimizations to make them usable and robust. Collaboration technologies available today largely target videoconferencing, webcasting, or shared whiteboard technologies and are expensive, closed systems often using proprietary protocols.

We advocate an approach that is based on open source infrastructure designed for easy integration into a wide variety of applications and environments. With this approach, even simple ideas could provide powerful new modes of collaboration that fit invisibly into the working environment and style of the research scientist. We have described three separate ideas that fit this model: (1) *the Vast Pixel Savannah* –the collaborative potential is built into the underlying abstractions applying to our display surfaces, (2) *Pixel Porting* – relies on a ubiquitous tool that lives side-by-side with any application, and (3) *IM Everywhere* – conveys collaborator status to even our most mundane of information sources by wrapping them in agent technology and opening an instant messaging channel to them. These are probably not the only options, but they are all designed into the working fabric of our computing environments well below (or outside of) the imaginary box containing an application. This property endows them with the power of independence making them always available.

ACKNOWLEDGEMENT

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357.

REFERENCES

[1] Johnson, C., R. Ross, S. Ahern, J. Ahrens, W. Bethel, K.-L. Ma, M. Papka, J. v. Rosendale, H.-W. Shen, and J. Thomas, "Visualization and Knowledge Discovery: Report from the DOE/ASCR Workshop on Visual Analysis and Data Exploration at Extreme Scale," Salt Lake City October 2007.

[2] Disz, T. L., M. E. Papka, M. Pellegrino, and M. Szymanski, "CAVEcomm Users Manual," Argonne National Laboratory, Argonne, Technical Memoramum ANL/MCS-TM-218, September 1996.

[3] Disz, T. L., R. Olson, M. E. Papka, R. Stevens, M. Szymanski, and R. J. Firby, "Two Implementations of Shared Virtual Space Environments," Argonne National Laboratory, Argonne, Preprint ANL/MCS-P652-0297, March 1997.

[4] Foster, I., M. E. Papka, and R. Stevens, "Tools for Distributed Collaborative Environments: A Research Agenda," in High Performance Distributed Computing Focus Workshop on Multimedia and Collaborative Environments,, Portland, Oregon, 1996.

[5] Stevens, R., "Group-Oriented Collaboration: The Access Grid Collaboration System," in The Grid 2: Blueprint for a New Computing Infrastructure, 2nd Edition ed, I. Foster and C. Kesselman, Eds.: Morgan Kaufmann, 2004, pp. 191-200.

[6] Olson, R., and M. E. Papka, "Remote Visualization with VIC/VTK," in Visualization 2000 Hot Topics Salt Lake City, UT: IEEE Computer Society, 2000.

[7] Karonis N., M. E. Papka, J. Binns, J. Bresnahan, J. Insley, D. Jones, and J. Link, "High-Resolution Remote Rendering of Large Datasets in a Collaborative Environment," Future Generation of Computer Systems, 2003, pp. 909-917.

[8] Binns, J., F. Dech, M. E. Papka, J. C. Silverstein, and R. Stevens, "Developing a Distributed Collaborative Radiological Visualization Application," in From Grid to HealthGrid, 2005, pp. 70-79.

[9] Binns, J., J. DiCarlo, J. A. Insley, T. Leggett, C. Lueninghoener, J.-P. Navarro, and M. E. Papka, "Enabling Community Access to TeraGrid Visualization Resources," Concurrency and Computation: Practice and Experience, vol. 19, 2006, pp. 783 - 794.