PicShare: A Capture-Centric Remote Collaboration Tool

Maurizio Pilu
Hardcopy Technology Laboratory
HP Laboratories Bristol
HPL-2000-149(R.1)
May 31, 2007*

remote collaboration, casual capture, imaging, scanners

Sharing images or documents is a natural activity between people both at work and in their social life. Sharing between remotely-located people is now possible through one of the many remote collaborations tools available on the market. However, none of them was particularly designed with image sharing in mind. This short technical report describes the principles of a prototype remote collaboration application we developed called PicShare that is tailored to sharing and discussing captured images of any kind. The concept of PicShare was developed to systematically satisfy some basic user needs when sharing material captured with a scanner or a digital camera. These needs are seamless integration with a capture device, a collaboration session encompassing a set of images, the selection images and their views and pointing, all without appreciable delay and assured consistency between the local and remote views. PicShare addresses all these issues in a single integrated solution that runs on the top of the ubiquitous NetMeeting ® videoconferencing application. Observation of PicShare usage by a small number of early users showed that their experience was greatly enhanced when compared to that with existing general purpose application-sharing tools. The integration of PicShare with a prototype desktop face-up scanner and its 2-seconds capture-to-share time has also stimulated thinking of the device itself as a remote collaboration tool, for instance through hand-written sketches or 3D objects that are captured and shared and discussed synchronously.

1. Introduction

Remote collaboration has become a fairly hot area in the past 10 years and ever more so since the widespread diffusion of the Internet in corporations and households. Some estimates put the potential value of distance learning and training market alone at $60 billion.

Remote collaboration has taken several forms to cater for different needs. Most large organizations have had room-based videoconferencing for years; these solution typically used proprietary technologies and were rather expensive to install and run. Recently, however, the convergence of data, video and voice to IP (Internet Protocol) is making solutions more affordable than ever and video conferencing standard have emerged, such as H323 and T120. Most of the action and dollars at the moment are on so called asynchronous *groupware* solutions, which allow groups of people, either geographically dispersed or co-located, to share a heterogeneous workspace. Groupware is a broad term meaning ‘the vendors and products that combine a number of collaborative functions, including message transfer agents, directory, gateways, e-mail clients, collaborative tasks and scheduling’ (quoted from [13]).

Here we are more concerned with the growing trend in *synchronous* remote collaboration (often called RTC, Real Time Collaboration) and videoconferencing, where the participants are allowed real-time interaction with their interlocutors. The camp here is very crowded too, with approximately fifty vendors currently chasing the market (Lotus and Microsoft being the biggest). There are data-only products, often service-based, and richer ones that include both voice and video. There are browser-based RTC solutions where the collaboration space lives in a WEB Browser, thick-clients solutions (where an application needs to be installed on both ends) and a mixture of both. For a comparisons and reviews please see, for instance, [3] or [9] but also the many books available on the subject.

The work described here was concerned with investigating optimal ways of sharing captured images such as that of a document, a magazine, a 3D object, or a holiday picture. The concept that was developed as a consequence aimed at systematically satisfying some basic user needs in this context. These needs were identified as *i)* seamless integration with a capture device; *ii)* a collaboration session encompassing a set of images; *iii)* the easy selection of images in this set, their view and pointing; *iv)* all without appreciable delay independently of network bandwidth; and *v)* with assured consistency between the local and remote views. The prototype that was born from these requirements, called *PicShare*, addresses all these issues in a single integrated application which uses the data services provided by the ubiquitous NetMeeting ® videoconferencing application. Whilst the PicShare application is being used, the users can benefit from the usual services provided by NetMeeting®, that is audio, video, chat, application sharing, whiteboard and file transfer.

2. The PicShare concept

The concept of PicShare was formed while experimenting with existing collaboration tools, in particular NetMeeting®, and by talking with colleagues about previous HP Labs endeavors in remote collaboration, such as DeskSlate [1,2].

Using a simple document camera prototype built from off-the shelf components [4], we started to experiment with the possibility of *instant* “sharing” of things and documents. There are several scenarios where this could be useful, such as families sharing a photo album or children artifacts, engineers sketching a diagram on paper to discuss it with a colleague, showing features of a PCB board, etc. All of them share the common need to share something in the physical world and can be captured pronto with a scanning device. This kind of sharing, of an image-based nature, is bound to happen in the familiar screen world at least until some sort of virtual/augmented-reality application will let users remotely interact effectively in the physical world as well [3].

Following Frohlich’s taxonomy [3], we can classify this sort of collaboration as “screen to screen synchronous collaboration” although, strictly speaking, the “synchronous” aspect takes place *after* the image is captured or brought into the sharing space (once and asynchronously). To emphasize this aspect, we have termed this kind of collaboration as *capture-centric*.

The PicShare concept was articulated to fulfill some simple needs in this capture-centric space. In the following we shall explain these requirements, by no means implying that these are the only needs one might have.

1. The path (time plus complexity) from capture to actually sharing must be as short as possible, ideally instantaneous. It is therefore necessary to minimize the GUI interactions needed from the moment one captures to the moment sharing is possible. Short set-up and data transfer times are also of key importance in a (typically) low bandwidth infrastructure. (Scenario. User1: “Please, let me show you this drawing I got!”).
2. A collaboration session might involve multiple captures and both users must have the ability to switch seamlessly between previously captured images during the course of their discussion. Switching between pictures must be synchronous and symmetric, that is both users should be able to switch back and forth between pictures and/or initiate another capture. (Scenario. User1: “Actually, let me go back for a moment to the previous picture!”; User2: “No, wait a sec! Let me show you this idea of mine instead.”).
3. Users must be able to have sessions with different people, some of whom they might have already remotely-collaborated with before. Therefore, we might need separate sharing spaces similarly for documents and pictures that were previously shared with a particular user or for different topics. Automatic picture synchronization can then be used to avoid negotiating who has got what.
4. Both users must have the ability of zoom in to particular regions of the picture and make sure their view is shared with the interlocutor. This is necessary because screen resolution is typically much lower than the resolution of the

captured image and its details might not be visible in full view. A text document is a good example of this. (Scenario. User1: “Let me show you this in more detail!”)

5. Both users must have the ability to point at a particular location of an image in order to facilitate their conversation and make sure that the remote user sees the pointer as well. (Scenario. User1: “This is X”; User2: “Oh yeah, and what is that?”).
6. In a multi platform environment, people can have displays of different sizes, resolution and bit depth. Images should look as good as they can be displayed on the local and remote machines and their display sizes should be view-based and not pixel-based[1]. Experience with T120-based applications sharing tools (such as NetMeeting™) clearly show that these basic requirements for sharing pictures were not part of the design specifications.

As we shall see, all these features have all been embodied into *PicShare* to produce a single image sharing solution.

PicShare is a peer-to-peer capture-centric image sharing application based on the simple concept of sharing views of and pointers to images that are stored locally on all the participant’s machines. Before being able to share a new image, the originator *sends* a compresse version of the image to the connected peer(s). The sent image is placed in the current repository of the remote machine, which is called the *briefcase.* From this moment on, sharing is enabled on these multiple local versions of these images; typical sharing actions, such as changing views and pointing reduce to low-bandwidth negotiation simple commands between the remote and local applications. The local and remote user’s briefcase may contains many images that may or may not be available on both machines. This state (shareable or not) is signaled for each image to the users. At any point in time a user can decide to add a new image to the briefcase by either importing or capturing it. A capture-to-share feature is supported that, when the connection is available, immediately shares (sends) the imported or captured image.

PicShare is also an application that is directly integrated with scanning devices. A general purpose application sharing tool is not integrated with scanning devices and recur to an image viewer that is shared as a whole application; the steps are typically initiating the scan, saving a scanned file, opening an application and sharing the application itself. In PicShare a scan-to-share paradigm is used where the captured image is directly available for sharing without the need of another application. This considerably simplifies the user interface and rid the user of intermediate steps.

[1] With *view-based*, we mean that a particular view in picture coordinates can be forced regardless the actual pixel resolution of the screen. For instance, if we want to display the top-left quarter of a 300dpi A4 document on the screen, in a pixel-based rendering we need 1000x1500 pixels on the screen, which is beyond most display’s capabilities. On the other hand, a view-based rendering would display the top-left quarter sub-sampled at whatever screen resolution and window size the remote user has available. For instance, with NetMeeting™ application sharing is pixels based and explicit negotiation of the screen capabilities is needed between remote and local users in order to make sure actual views are truly shared.

PicShare allows for sessions that require multiple pictures, captured on or off-line. All the pictures for the session are held in a local repository called briefcase. Users can then switch back and forth between these pictures without needing to search and open files into the image viewer and close older ones.

In PicShare, users can switch between briefcases or create new ones, but it is within a briefcase that multi-picture collaboration becomes seamlessly easy. In fact, at any moment in time, the connected users know which picture could be shared in their current briefcases and can select one of them to share with virtually no delay. We call this feature “Briefcase Synchronization” and has been adapted from, for instance, the Briefcase folder of the Windows® operating system. One user might have a briefcase that is used for all his sharing sessions while another, more organized, could have a briefcase for each topic or project; nonetheless, when connected to each other these two users are at any moment aware of what document is available at the remote end. It is up to individual users to decide whether to share other pictures that are not currently shareable or capture a new one to be immediately shared.

When a user has captured or selected a picture for sharing in PicShare, both users can select areas to magnify, or highlight things during their discussion. Since the pictures are stored locally, changing view, zooming, scrolling, etc, does not involve sending actual pixels and the result is that even on extremely slow connections the reflected change on the remote application is virtually instantaneous. Application sharing tools such as NetMeeting® (see Appendix 1) are based on a very general paradigm but unfortunately are extremely slow at transmitting changes with slow connections (even with corporate networks the delay is measured in seconds). In addition, a pointer feature is provided that gives users an indication of what the user at the other end is referring to. Given that this pointer is not the same as the mouse of the system GUI (as in standard application sharing tools), some extended pointer features are possible.

PicShare is wholly display-size and bit-depth independent. This features is a consequence of the images being rendered locally rather then remotely, as in the case of NetMeeting® for instance. In PicShare the viewing conditions are always the best the current display can manage. It is possible for instance to have a user using a 1600x1200 screen with a remote user on a 800x600 screen and yet be sure that a local view matches the (scaled) remote view.

None of these features are truly novel, as they can be found scattered in several sharing applications on the market and even in the patent literature. For instance, HP Labs’ DeskSlate was “a pen-based telephone accessory which used a voice and data modem to provide a shared document workspace for telephone partners” but “lacked an integral scanner for easy document input [and users] were obliged to fax to their own device before making a call” [3]. The ill-fated appliance had file synchronization, instruction-based view management and other interesting features at the time. Clearly local-remote file synchronization in PicShare’s briefcase is possible by many groupware solution (e.g. Lotus’ Domino®). Also sharing local and remote views through high-level commands is, for instance, well articulated in an Intel patent [10]. And the list can go on.

Nevertheless, PicShare brought all these ideas together coherently to make the image sharing useful and easy-to-use.

3. PicShare Implementation

This section is devoted to the description of the actual technical details of the PicShare prototype (v1.0), specifically its user interface the implementation strategies. The description of the user interface should hopefully help clarify the description of the concepts and features given in the previous section. (Note that the current version does not allow for more that two people in a session.)

3.1. Functionalities and User Interface

Figure 2 shows a screen shot of PicShare 1.0 in action that we shall use to describe some of the features.

The application window is composed of five areas. At the top we have a toolbar where the most common features can be accessed directly. On the top left we have the briefcase (1. Below there is the preview pan (5) that is used to see the location of the current zoomed view and let the user drag the view as well. The biggest region is the main view area (4), which is where most of the picture sharing action takes place. The strip at the bottom is a status bar containing various information.

The briefcase (1) is where a user keeps all the images files s/he would like to use in a session. Different icons beside the image names indicate those that are currently shareable (i.e. present locally on both machines) and those that are not. We call this the synchronization status. Scrolling the image list or clicking on a file causes the corresponding image to be displayed in both the preview and main display areas; if the connection is active and the selected image file is shareable, the new view will also immediately appear at the remote end as well. As well the usual file operations, the briefcase toolbar (2) is used for two purposes, sending a file for sharing and refreshing the synchronization status. The former is used when a image has been captured with PicShare or has been imported into the briefcase and the user decides to share it immediately with the remote (and currently connected) user. The latter is used to refresh the synchronization status of each file in the briefcase (not strictly necessary).

Another briefcase, perhaps referring to another project or user, can be selected at any time on or off-line using the briefcase selection button (3). A pull-down list conveniently allows speedy selection of recently used briefcases. Currently, the briefcase is nothing more than a directory, so one could select an existing directory in the file system and make it become the current briefcase.

The main view (4) and preview pane (5) allow the user to select regions to zoom in (by selecting an area with the mouse), zoom out and scroll. These two views are synchronized and the preview pane is used to have a overview of what region the main view is focused in. The red outlines (6) and (7) are very important because, regardless what the windows actually show, they indicate what it is *guaranteed* to be visible at the remote end. In some sense it is this that is the truly shared area.

Finally, the two scanning buttons (8) are used to scan either with a Twain compatible scanner or with our prototype desktop camera [4]. Note that in both cases capturing is a very simple one-button operation that uses a default and optimal setting[2]. When capturing with our prototype desktop camera [4] the main view (4) is temporarily replaced by a *live preview* window that allows the positioning of the subject, should the users wish to. Once captured the image ends up directly in the briefcase and the user can initiate the collaboration by pressing the "sharing" button on the briefcase.

3.2. The implementation of PicShare

The most important architectural choice that was taken during the early design stages was to use NetMeeting® [5] as the communication platform that would support audio and video and data conferencing. In Microsoft® own words, the "NetMeeting® 3 application programming interface (API) provides strategies, technologies and tools that can help you build Internet conferencing applications".

This choice offered tremendous advantages. Through its COM API, NetMeeting® offers services such as the Node Management and Call Control, Audio and Video using Real Time Protocol. NetMeeting™ is fundamentally based on two widely accepted standards, the ITU H323 standard for the audio/video part and the ITU T120 standard for the data part, which includes application sharing and file transfer. Microsoft® provides the developer community with a free (but far from easy to use) Software Development Kit called the NetMeeting® 3 SDK [6]. This SDK uses COM interfaces to access many of the features of NetMeeting® along with those of the H323 and T120 standards.
This architectural choice implies that PicShare does not *per se* handle the connection or deals with audio video communication but rather it is an application that piggy-backs onto an existing NetMeeting® session between two users and uses a dedicated data and file transfer channel for its operations.

[2] Future versions will allow interactive setting of defaults scanning parameters.

Figure 1 shows an overview of the architecture of the PicShare application. The development has been carried out using as much as possible a component-based philosophy. The grayed out outline encloses modules that constitute the PicShare application but are, effectively, independent components. The top-level part of the application has been written in Visual Basic 6.0 [8], which is well suited for this sort of application.

One component is the NetMeeting® control component, which deals with creating an easy to use and clean interface with the NetMeeting® COM API. This component has been implemented as a COM Control (COM object which implements the IDispatch interface) using Visual C++ 6.0 [7] and can therefore be included in a Visual Basic application. The component, amongst others, implements two methods, SendData and SendFile, and generates two events, DataReceived and FileReceived; these four are the backbone of all the communication between two PicShare applications over the NetMeeting® conference.

Another component, ViewSync, deals with synchronizing views and other things (such as pointers) between two PicShare applications. This component, implemented in VisualBasic 6.0, abstracts details of the actual viewing component and the communication mechanisms.

Next, there is the viewer component. The viewer component currently used by PicShare is based on the Kodak Imaging (or Wang Imaging in some systems) component that is installed on every Windows 98/NT/2000 machine. However we are planning to re-implement this component from scratch because we encountered several problems with it, such as incompatibility with recent JPEG or TIFF images, and difficulties with rendering overlay graphics on it.

The “desktop camera” component is the one that has been implemented to interface the application with the prototype desktop camera [4]. This was implemented in Visual C++ 6.0 and maps almost one-to-one the services offered by the Scanner API supported in [4].

Finally there is the “briefcase” component. The local and remote briefcases exchange messages to each other to negotiate which image files are available via messages and methods interfaces with the high-level part of the application which in turn calls the communication component. The briefcase was implemented as a control in Visual Basic 6.0.

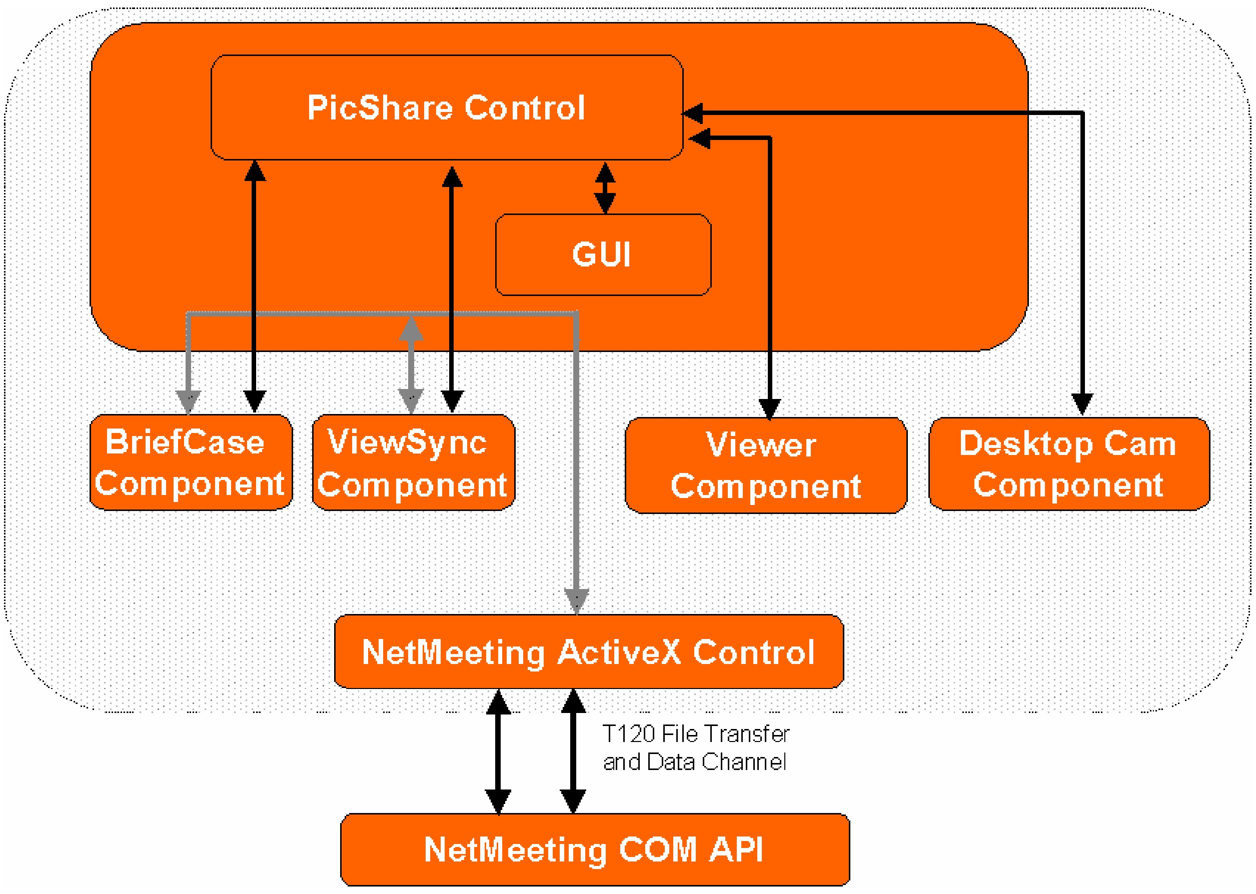

Figure 1 The PicShare 1.0 architectural overview.

The only part of the application that needs to be aware of what the other components look like is the PicShare control, which orchestrates the inter-component interactions and how they affect or are affected by the user interface.

One last remark. Given that there is only one data channel available but there are several entities in the application that need to interact, a simple messaging system based upon command strings has been implemented that allows, for instance, the local briefcase to talk to the remote briefcase, whilst the ViewSync component is doing its view synchronization job with its counterpart.

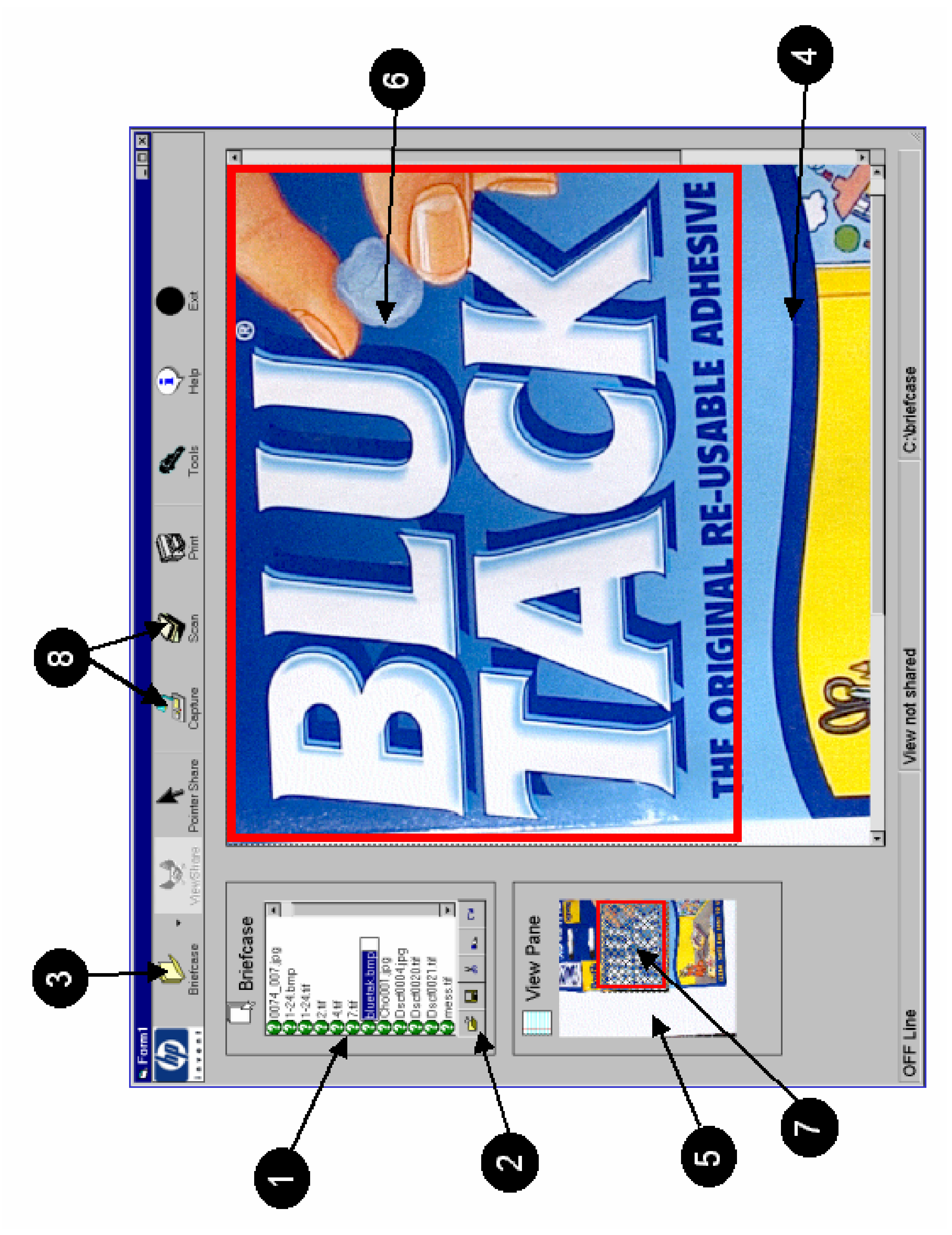

Figure 2 The PicShare interface.

4. A bit of perspective

In previous sections we have discussed the aspects embodied in the PicShare concept and the reasons as to why we thought they were a step forward from general-purpose conferencing applications. Not surprisingly, informal feedback from early users showed that the usability and experience is greatly enhanced and more consistent with expectations.

But a few questions need to be answered. First, how relevant is PicShare in the Real-Time Conferencing (RTC) panorama? And how novel are its features. What are the current alternatives in the market? And what is the HP's business perspective in all this?

Despite the crowded RTC panorama, the proposition of a vertical product such as PicShare is one of a solution dedicated to be a good companion to a capture device for image-based collaboration.

Many hypothetical situations suggest that a capture-centric solution is better than a general purpose one. For instance, as argued by Frohlich in [3], paper is still the preferred medium to aid articulating concepts and sketching things and a specialized solution makes the transition from physical to the shared digital space very easy. When comparing with a standard way of sharing (see Appendix), the advantages are all too evident. The integration of PicShare with our prototype desktop face-up scanner [4] and its two-seconds capture-to-sharing time has even further highlighted the need for a specialized solution to make the most of the new scanning paradigm. The extremely low time delay when actually sharing an image or switching to another one has also impressed all people that had previous experiences with other desktop collaborations tools.

Furthermore, the live preview available when capturing with our prototype desktop camera [4] has proven invaluable to show the potential of face-up scanning together with a sharing tool such as PicShare, opening up exciting possibilities of a smooth link between the physical world and a shared digital workspace.

Some products available on the market have addressed successfully vertical applications within the larger remote collaboration scenario, and PicShare resemble some of them. The most notable one is WebEx®'s *collaborative browsing* feature [11]. It allows user to synchronously share a web browsing session; this is not done by sending the browser window's pixels (as in standard application-sharing) but through high-level commands. Probably this is clearly a superior solution to application sharing since it can support a far more efficient collaboration experience over a low-bandwidth connection and it is platform and display independent.

If we can draw from this, it is the knowledge of the shared object that allows efficiencies through the definition of high-level sharing actions. In the case of WebEx® an action could be "Connect to www.hp.com" whereas in our case it is "View 100,100,800,300" or "Display Pic1.jpg". The T120 specification onto which the NetMeeting® data services are based does support high level commands for image annotation for whiteboards and the likes and its application sharing feature could not be more comprehensive, allowing total remote control of any application. But it is especially the *use model* that is not optimized for an image sharing application. As far as image sharing is concerned, *PicShare is to NetMeeting®'s application sharing what a Photo organizer is to Internet Explorer® for the task of organizing a personal photo album: just a more* appropriate *use model*.

On the business side, HP is lately showing interest in the lucrative remote tele-working market. Recently, HP has made an equity investment in TManage[3] (www.tmanage.com) as part of a new strategic alliance aimed at delivering products and services to the emerging corporate/home office (COHO market). HP has also made a strategic investment in PlaceWare, a major conference vendor; HP later released a product based on PlaceWare as a full-blown web-conferencing service for the education market called Jroom [3]. Reading between the lines, we can anticipate that HP has recognized the huge growth potential offered by providing services and technology to the millions of teleworkers.

HP has also got into on-line photo albums with Cartogra (www.cartgogra.com) as a vehicle to promote its imaging business and provide various image services to its customers. Cartogra allows sharing in a simple asynchronous fashion being mainly a repository for images. Without entering the discussion on whether on-line photo albums are founded on a viable business model, it is a short leap of imagination to think as Cartogra as the ideal platform to launch a browser-based on-line image-sharing tool with functionalities similar to PicShare's. In this respect, PicShare more sophisticated successors could also evolve towards a thin-client model, whereby a Java enabled web browser could run a sharing applet that could support sharing images stored on a server, on the web or locally.

Within HP, the future of a vertical, capture-centric image sharing tool is probably intertwined with that of the scanner business (either flatbed or document camera) and with technology integration issues. At the moment, PicShare is a full-blown application and wholly replaces the main scanner application (currently PrecisionScan) in a capture-and-share session. However the sharing feature could be more tightly integrated with the main scanner application; this avenue will need to be explored further both in terms of technology compatibility and usability issues (such as consistency of GUI and paradigm shift).

Concluding, it is wiser to consider capture-centric image sharing as a part of a larger HP solution in the area intersecting casual capture and remote collaboration then might evolve naturally from the existing product portfolio.

[3] TManage in one of the market leaders in the sector who provide telecommuting situation audit, telecommuter profiling assessments, technology deployment, maintenance and support (www.tmanage.com)

Acknowledgements

Thanks to Stephen Pollard and David Slatter for contributing to the concepts embodied in PicShare and David Frohlich for his constant input of ideas and references. Thanks also to Andy Hunter for discussions on HP Labs' DeskSlate product. I would also like to acknowledge Andrea Zambianchi as a contributor to the SW implementation and the solution of some technical problems.

References

1. O'Conaill B., Geelhoed G. & Toft P. (1994) Deskslate: A shared workspace for telephone partners. *Companion Proceedings of CHI '94*: 303-304. New York: ACM Press.
2. Hunter A. (1995) Pen-based interaction for a shared telephone workspace. Chapter 5 in J. Nielsen (Ed.) *Advances in human-computer interaction Volume 5*. Norwood, New Jersey: Ablex Publishing.
3. David Frohlich (2000). Beyond remote collaboration: Opportunities for audio-visual support of interpersonal collaboration. *HP-Labs Technical Report*, In Preparation.
4. Pilu M. & Pollard S. (2000). A Prototype Desktop Document Camera. *HP-Labs Technical Report*, In Preparation.
5. Microsoft(1999). NetMeeting 3.0. www.microsoft.com/windows/netmeeting.
6. Microsoft (1999). The NetMeeting 3.0 SDK. www.microsoft.com/windows/NetMeeting/Authors/SDK
7. Microsoft (1999). Visual C++ 6.0, msdn.microsoft.com/vstudio/
8. Microsoft (1999). Visual Basic 6.0, msdn.microsoft.com/vstudio/
9. Collaborative Strategies (1999). The Data Conferencing and Real Time Collaboration Market in the New Millennium. www.collaborate.com.
10. Porter, D. *et al* (1998). Method and Apparatus for synchronizing viewed information in a conferencing environment, US Patent US 5,740,161, Intel Corporation.
11. ActiveTouch Co. (1999). The WebEx Meeting Center, www.webex.com.

Appendix. Application-Sharing with NetMeeting™

This appendix overviews the application-sharing features of NetMeeting®. These features are based on the ITU T120 standard as with many other similar tools available on the market.

Rather than paraphrasing, here we quote Microsoft's description of the feature, acknowledging Microsoft copyright (underlines are the author's).

> "Shared programs allow meeting participants to view and work on files simultaneously. For example, you may have a Microsoft Word document that several people need to work on. You can open the document on your computer, share it, and then everyone can provide their comments directly in the document. Only the person who has opened the file is required to have the program on their computer. Other participants can work on the document without having the program. Only one person can be in control of a shared program at a time. If controllable appears in the title bar of the shared program window, the person who shared the program has control and is allowing others to work in the program. If the mouse pointer has a box with initials, then another meeting participant has control of the program. All meeting participants can share programs during a meeting. The shared programs of each participant appear in separate shared program windows on the other participants' desktops".

NetMeeting® is as very versatile tool for remote collaboration[4]. However the underlined parts of the above quote highlight some of the features that make it not ideal (if not cumbersome) for a capture-centric image-sharing application. First we see that "files need be open in an application and the application shared". Then we see that control is asymmetric, that is there is a negotiation to establish who is in control: two participants cannot freely interact. Finally we see one of the greatest problem, the concept that *windows* are shared, not images; if one needs to share a document that was just scanned, s/he has to share a window that contains his/her viewer application, and the document itself. This leads to problems of view and color incompatibility and excessive use of bandwidth. And these are only few example. Anybody can carry out simple usability experiment using the NetMeeting® installed on their Windows® machines.

It is clear that its versatility has turned (as often is the case) to a disadvantage compared to a tailored application which does few things but very well.

[4] The author himself has successfully done some real-life remote support and collaboration sessions with colleagues and was impressed by the usefulness of it. It is opinion of the author (and not necessarily HP's) that reason why NetMeeting™ is freely available has more to do with Microsoft's business strategy than the intrinsic values of the product. In fact once NetMeeting™ started to be distributed for free, most of the competition moved from a *product* model towards a *service* model with added for their proprietary collaboration technology.