

Microsoft First Contact: The Borg Experiment

Charles Herring

Department of Computer Science and Electrical Engineering
The University of Queensland, St Lucia, 4072 QLD, Australia
herring@dstc.edu.au

Abstract

We describe our experiences in developing a Computer Supported Collaborative Work (CSCW) prototype based only on Microsoft software. We undertook this experiment to gain insight into exactly how the various Microsoft infrastructural components and application packages could be used in developing collaborative systems. The paper contains a brief overview of the technologies we explored. Based on the knowledge gained during this first phase of the study, we decided upon the functional requirements for the prototype. Sometime during this period our coworkers began calling us “the Borg” after the Star Trek characters. We accepted this and developed our demonstration based on the Borg theme. We describe in detail the implementation of the prototype and show screen dumps of the running system. We make extensive use of the windows-browser integration to create a domain specific environment. A novel aspect of our system is how we provide “presence and awareness” by representing users (Borg) in the file system as file short-cuts. We conclude with some reflections on our experiment and provide thoughts on how future CSCW systems will be implemented using the Microsoft platform.

1 INTRODUCTION

We report on a “technology experiment” to develop a prototype Computer Supported Collaborative Work (CSCW) system based solely on the latest Microsoft software. Our team, the “wOrlds” group at DSTC, conducts research in CSCW and has developed several prototype systems [1]. These prototypes have been developed based on a traditional academic, computer science software approach: Unix, GNU, shareware and freeware. Some commercial packages are used when necessary. Of course “platform independence” is a requirement: the systems must run on both Unix and Windows. With the proliferation of Microsoft, we decided it was time to investigate how a CSCW environment would be developed using *only* Microsoft software. Additionally, our aim was to implement some of the advanced features that characterize our Locales theory [2].

Our first task was to gain familiarity with the (bewildering) array of Microsoft offerings. This was necessary to determine which products should be used, their limitations, and to make informed judgements about implementation issues. The components and systems we explored included: Microsoft infrastructure (COM/DCOM/ActiveX), the new standard Dynamic Hypertext Markup Language (DHTML), the NetMeeting software development kit, Microsoft Exchange, and the integrated Windows-browser (Internet Explorer 4.0, Active Desktop and Channels) environment. A brief overview of these technologies is given in Section 2.

During the process of learning about the Microsoft collaboration software, our fellow team members began, jokingly, referring to us as “The Borg” after the Star Trek characters. (Note: the term Borg is used to refer to an individual or a group.) We happily accepted the idea of having been “assimilated” by Microsoft as it was in keeping with the spirit of our experiment. We also recognized the advantage of having a domain specific setting for our prototype system. We realized that using the Borg theme would permit the development of coherent scenarios for demonstrating the features of the system. Therefore, we designed and built our prototype around the Star Trek/Borg characters.

Having gained sufficient familiarity with the Microsoft platform, we were in position to design a prototype CSCW environment. Section 3 describes the functional requirements we chose and section 4 goes into detail of the design and implementation of the prototype. Section 5 shows the working system based on a usage scenario. We conclude with lessons learned and reflect on the experiment

2 OVERVIEW OF MICROSOFT TECHNOLOGY

The first step was to immerse ourselves in Microsoft technology. Much of this period was spent browsing the Microsoft web site and downloading everything we thought would be useful. It was a bottom-up approach aimed at getting a broad overview of the vast range of products they have produced. We looked at the underlying technologies of COM/DCOM/ActiveX, and DHTML and the Active Desktop, as well as higher level tools including NetMeeting and Exchange.

Microsoft's evolutionary approach to software development often leads to confusing product naming. This is especially true for the COM, DCOM, OLE and ActiveX family. COM, the Component Object Model, is Microsoft's component framework. It is a binary standard for building and running component-based applications. COM objects encapsulate functionality and provide standard interface support for communication among objects. As a binary standard, COM objects are language independent and can be used and extended without having to manipulate their original source code. COM is inherently distributed and DCOM, Distributed COM, is just COM used in a distributed fashion. OLE is an extension built on top of COM for handling compound documents. ActiveX is another extension to COM that provides a wide range of functionality. ActiveX "controls" include low level support for basic Windows user interface features such as menus, buttons, drag-and-drop, etc. ActiveX is also Microsoft's answer to Java in terms of Internet downloadable programs. ActiveX controls can be embedded in HTML pages and automatically downloaded. We found many vendors are now using COM and ActiveX to restructure their applications and make them more open and flexible by providing them in component form to developers. Visual Basic is an easy way to learn how to program with ActiveX and we made use of it in our prototype.

Microsoft's integration of Windows95 and Internet Explorer 4.0 has certainly been in the news and courtrooms lately. The merging of the desktop with the web-browsing paradigm removes the distinction between remote and local information. For example, local file folders are viewed using the browser and may be customized by editing their associated HTML pages. One feature of the windows-browser integration is called the "Active Desktop." The Active Desktop consists of three layers. The bottom layer permits arrangement and display of items specified by URL. (This replaces the older "wallpaper" layer.) The second layer is the traditional "icon" layer and the third layer is where running Windows applications reside. Items such as ActiveX controls, images or HTML documents may be placed on the bottom layer underneath icon layer. Another feature is Microsoft Channels displayed as a menu bar on the Active Desktop. Channels permit easy access to web sites of particular interest. With channels enabled, specified web sites are periodically checked for changes and the user is notified or information is automatically downloaded.

Another major piece of Microsoft technology we learned about is Dynamic HTML (DHTML). DHTML is the glue that holds all the other components together. It is an extension of HTML to include an object model that provides more control and functionality than HTML. The object model provides a container hierarchy through which events may be passed – bubbled up to higher layers. By exposing the properties, methods and events of each HTML element on a page, greater control is permitted. This allows for precise 2-D layout, a Z-axis dimension, and cascading style sheets. Multimedia, ActiveX, Java applets, JavaScript and VBscript can interact and be controlled through the DHTML object model.

The two packages we spent the most time with are NetMeeting and Exchange. NetMeeting is a teleconferencing and applications sharing package. It contains audio and video, application sharing, shared whiteboards, chat and file transfer. The NetMeeting SDK (Software Development Kit) is a good example of how a complex application is built using COM and ActiveX controls. It is supplied with examples that show how to build applications using Visual Basic, C++ and ActiveX components in DHTML pages. An Internet Locator Service (ILS) program is used by NetMeeting to connect parties who wish to meet. We found many third party companies developing products based on NetMeeting and ILS.

Exchange is Microsoft's messaging and GroupWare enterprise platform. It is their answer to Lotus Notes. The Exchange Server provides email, shared folders for scheduling, calendars, journals, address books and to-do lists. It is accessed by client software such as Outlook and by web browsers as well. It can be customized for specific applications through use of "Collaborative Data Objects" facility. Exchange

supports many standard protocols such as SMTP/POP3, IMAP4, HTTP, LDAPv3, and NNTP that permit interoperability with other systems. It has a host of features, such as the ability to schedule NetMeeting appointments to start automatically.

3 FUNCTIONAL REQUIREMENTS

Having completed exploration of the technologies described above, we were in position to decide what functionality could feasibly be demonstrated by the prototype. Based on the functional requirements, we evaluated several design and implementation approaches. A goal of the effort was to write as little software as possible and to use as many of the customization features inherent in the Windows-browser integration (e.g., IE4, Active Desktop) as possible. The detail of how we developed the prototype is the subject of this section.

CSCW, as a discipline, exists at the interface between human work groups and computer technology. In this regard it seeks to incorporate a range of complex and subtle aspects of human activity. The difficulty in designing a one-size-fits-all CSCW system lies in the variety exhibited by individuals and the groups, communities and organizations they form. Such a system must either anticipate all the features that might be required or provide for a high degree of customization. However, over the last decade a number of commonly used tools have evolved and proved useful (and are now expected) in any networked computer environment. Microsoft has, understandably, developed a number of these products to address basic user requirements. Therefore, we sought to make use of as many of these readily available products as practicable in our system.

However, to ensure a CSCW system will meet user's needs, those needs must be understood. Hence some analysis of user requirements must be attempted. Surprise! Given any non-trivial domain, there are user requirements that will not be met by a generic package. In our case The Borg have a unique form of distributed-shared memory. Every Borg knows where every other Borg is, what they are thinking, and what they are doing. Further the Borg possess a collective understanding of long-range goals that result in them behaving as an organic, self-organizing system. (Really, the Borg aren't so bad. They don't intend to harm. They only want to assimilate people to a higher cause.)

Therefore in addition to providing the commonly available collaborative support facilities, we were faced with development of a challenging new form of work group support. We thought it important to provide the Borg with a high fidelity mapping between their conceptual understanding of the real world and their work environment. Knowledge of the global workspace would be necessary in order to provide the Borg with an evolving, situational awareness of common goals. The Borg would require means to gather and disseminate information rapidly. Each Borg would need to have his own workspace as well as access to the workspaces of all other Borg. They would require the ability to share any type of work product or artifact in the workspaces. Also, the Borg would need a dynamic picture of the movements of the Borg through the workspaces. We needed to provide a convenient way for them to instantly know where any Borg was and immediately contact him by a variety of means. Easily established audio and video connection was seen as a necessity. Finally we wanted to provide the Borg with intuitive and familiar symbols and icons so they could readily identify aspects of their environment.

We settled on the following software capabilities as requirements to support the Borg prototype. They are summarized below grouped into four categories after Wilson [3]:

1. Communications mechanisms:
 - Synchronous: audio, video and chat
 - Asynchronous: email, presence, awareness and notification
2. Shared work space facilities:
 - Hierarchical work spaces
 - Whiteboard
 - General application sharing

3. Shared information facilities:
 - Shared file system
 - Hypertext and hypermedia
 - Channels

4. Group activity support facilities:
 - Group calendars
 - Task management
 - Meeting scheduling
 - Event journaling

4. DESIGN AND IMPLEMENTATION

Now we describe the design and implementation of our prototype and how we used Microsoft software to realize the above capabilities. But before going into those details, we record here the hardware and software environment used to develop and demonstrate the prototype.

First we describe the hardware and software environment available to build the prototype. For hardware we had three generic PCs with 166MHz Pentium MMX processor, 128MB of RAM, 3.2 GB hard disk, 100 Mbps network card, Winnov/Videum (See <http://www.winnov.com>.) audio/video capture card and camera, microphone/headset, and 19" (1600 X 1200 dpi) monitor. As we subscribe to the Microsoft Developer's Network, we have all of Microsoft's packages. We set one machine up as a Windows NT4.0 domain server so that we could run Exchange 5.5. The other two machines were Windows NT4.0 workstations within that domain and they mounted a shared drive from it. We also downloaded and installed the IE4.0 beta to get the latest Windows-browser integration features. The NetMeeting 2.0 SDK was to play an important role in the prototype as was Visual Basic 5.0 for developing ActiveX controls. We also made use of the Winnov/Videum ActiveX controls for video capture.

Providing a shared workspace to support the particular collaborative community of the Borg was the major design challenge. Our "Prime Directive" for acceptance of any design alternative was that it *did not require writing code*. We imposed this constraint because of time limitations, but more importantly because we wanted to determine how flexible the integrated Windows-browser approach really was. We came very close to achieving the goal of not writing any code. We ended up writing four ActiveX controls in Visual Basic and customizing a few DHTML pages. That was the extent of the code development. The remainder of the effort was in setting up the packages, the file system structure, tailoring the displays with various images and icons, and finding props for the demonstration scenario. We now describe in some detail how we accomplished meeting the functional requirements stated above.

A major design decision was the use of folders (directories) on the shared file system as the main shared information facility. While it can be argued this imposes a scalability limit on the system to work only on local area network with shared drives; it greatly simplified development of the prototype and provided for a high degree of flexibility. This decision permitted the use of the Windows-browser integration to tailor the folders to meet a large number of the user requirements. What the Windows-browser integration means, from the user's point of view, is that there is no distinction between browsing the local file system or accessing a DHTML page on a remote server. What this means from a software point of view is that the same program (i.e. `explorere.exe`) is used to inspect the local file system folders as well as navigate the web. So in fact, our choice to use the shared file system approach is quite extensible to the wide area case and functionally equivalent from a software perspective. As this is very important to understanding the implementation, we explain precisely how this works.

Under Windows95 and NT4 (with IE4 installed) if a folder contains a file named **Desktop.ini**, the instructions in that file will be processed by the operating system when the folder is opened. These instructions determine how the folder will be presented to the user. This feature permits customizing the behavior of folders. Figure 1 shows the content of a typical Desktop.ini file used in our system. The **PersistMoniker** field specifies a file URL to be loaded (into the browser – `explorere.exe`) when the folder

is opened. That HTML file can be tailored to perform whatever functions are needed. We use this capability, along with the **IconFile** and **IconArea_Image** fields, to build domain specific shared workspaces. To see how this works, open a folder and select the menu options *View->Customize This Folder*. This will bring up a Wizard that eventually lets you edit the HTML file specifying the behavior of the folder. For example, in the default HTML file there is an ActiveX control, "FileList", that displays the contents of the directory.

```
[ExtShellFolderViews]
{5984FFE0-28D4-11CF-AE66-08002B2E1262}={5984FFE0-28D4-11CF-AE66-08002B2E1262}
{BE098140-A513-11D0-A3A4-00C04FD706EC}={BE098140-A513-11D0-A3A4-00C04FD706EC}
Default={5984FFE0-28D4-11CF-AE66-08002B2E1262}
[[5984FFE0-28D4-11CF-AE66-08002B2E1262]]
PersistMoniker="file:///z:/the borg collective/repository/pages/Disseminator.htm"
[ShellClassInfo]
ConfirmFileOp=0
IconFile=z:\The Borg Collective\Icons\Ship Icons\Borg-3.ico
[[BE098140-A513-11D0-A3A4-00C04FD706EC]]
Attributes=1
IconArea_Image="z:\the borg collective\images\space.gif"
IconArea_Text=0x00FFFFFF
```

Figure 1 Desktop.ini

We used the ability to present the user interface via a DHTML/ActiveX specification to bring together a number of desired capabilities. These include the special features of presence and awareness as well as to provide access to other shared group activity support facilities. A novel feature of our system is in the implementation of "awareness and presence" that we now describe.

We decided to implement awareness and presence using the shared file system directly. Individual Borg would be represented as files, specifically, "short-cuts" to individualized DHTML files. These short-cut files, one for each user, are copied into the workspaces (folders) as the user moves around in the shared file system. The underlying Windows infrastructure provides for automatic notification (page refresh) to any other browsers when these files are copied into the folder so users immediately see when another user enters a workspace. This was programmed by including an ActiveX control in the DHTML file (loaded via the Desktop.ini file as described above). We had to write the ActiveX control that copied the short-cut files (pointing to the user's DHTML representation).

This ActiveX control is quite simple and is named **BorgPresence**. When a user opens (navigates to) a folder, the Desktop.ini is processed by the browser and this DHTML file activates the control. The control looks in the user's environment variables to find the path to his short-cut file and copies it into the folder. The Windows notification system automatically refreshes the browsers of other users so they see the presence of the person that "moved" into the folder. When the user leaves the folder, the short-cut file is deleted. (We also wrote an ActiveX control named **BorgEnvironment** that can search the user's environment variables.)

Another of our design goals in developing the prototype was to get away from the idea of users needing to go into a "collaboration program" in order to work together. We wanted there to be no distinction between the normal tools and work spaces and collaboration support capabilities. The short-cut file representation of a user serves this purpose by providing a convenient access point for establishing communications. The target of the short-cut file is the DHTML page that contains the audio/video functions for that user (i.e. the user represented by the short-cut.) We used components from the NetMeeting 2.0 SDK to implement the audio and wrote our own ActiveX controls to do the video.

The NetMeeting 2.0 SDK contains examples of how to use the various components to build custom conferencing applications. The examples range from using DHTML to coordinate ActiveX controls, Visual Basic, and C++. We chose to use the simplest features available from the SDK due to time constraints (and

the Prime Directive.) We used the Microsoft ILS (Internet Location Service) running on the server and the NetMeeting conference manager (conf.exe) on the clients to establish the audio connection. These can be accessed through a supplied ActiveX control running in the DHTML page. However, NetMeeting 2.0 SDK does not provide ActiveX controls for video. They provide a C++ example that uses video, but we did not want to spend the time to write our own ActiveX video control within the NetMeeting architecture. We describe our solution to providing multiple video sessions next.

We decided to use the ActiveX control supplied by the Winnov/Videum company to write a Visual Basic program to store the video from the camera. Figure 2 is a listing of the program and it uses two ActiveX controls. **WnvVideo** is the supplied by the Winnov company and **Timer** comes with Visual Basic. This program sets up a timer that saves the video frames to a file. The program is stored as an executable (videout.exe) and runs when a user logs on.

```
Dim CaptureFile As String

Public Sub StartVideo (Directory As String,
                      UserName As String,
                      Rate As Integer)
    WnvVideo1.Startup
    WnvVideo1.Enabled = True
    CaptureFile = Directory + UserName + "video.bmp"
    Timer1.Interval = Rate
    Timer1.Enabled = True
End Sub

Private Sub Timer1_Timer()
    WnvVideo1.GrabFrame
    WnvVideo1.CaptureFrameToFile (CaptureFile)
End Sub
```

Figure 2 Visual Basic code for Video Capture

An ActiveX control was needed to display the video frames from within a DHTML page. The code for this control is shown in Figure 3. It uses the **Timer** and **Picture** controls supplied with Visual Basic to display the video frames.

```
Dim CaptureFile As String

Public Sub StartVideo (Directory As String,
                      UserName As String,
                      Rate As Integer)
    CaptureFile = Directory + UserName + "video.bmp"
    Timer1.Interval = Rate
    Timer1.Enabled = True
End Sub

Private Sub Timer1_Timer()
    Picture1.Picture = LoadPicture (CaptureFile)
End Sub
```

Figure 3 Visual Basic code for Video Display

We call these shared workspaces, built by customizing the Desktop.ini and DHTML files together with the presence and awareness features, Borg Zones. They form the *centers* of work activity . [4]. Each Borg has his own personal Zone. Other specific Zones are established as needed depending on the goals of the collective. The Zones serve as the focus for group activity support features provided by Exchange 5.5. These include email, shared calendar, scheduling, task management, and event journaling. Each Zone has a corresponding set of shared Exchange folders set up to access those functions. The customized DHTML page for the Zone is used to give a single, consistent, means of accessing the Exchange server. Links are displayed on the Zone's page and clicking on them will bring up the selected folder in Exchange. Access to

other programs such as email and whiteboard are added in this way. This approach is in keeping with our design goal of seamless collaboration access within the common Windows-browser environment.

Other features of the Windows-browser integration used are the Active Desktop and Channels. The Active Desktop permits any URL specified item to be placed on the Windows background. We put the root folder of the Borg Zones on the Active Desktop along with several other support tools. These tools include a Borg Locator that shows the current location of the Borg by Zone and a channel bar with the Borg News Network. Using the Windows Display Properties we customized various aspects such as the background. We also added domain specific images and icons to further tailor the environment. The next section describes the demonstration scenario and has some screen dumps of the running system.

5 THE BORG DEMONSTRATION SCENARIO

Demonstrating a CSCW system is difficult. We have found that a definite scenario with detailed scripts and props, much like a play, is effective. Therefore, we developed a scenario consistent with our Star Trek theme called "Earth Assimilation." There are three Borg and the basic plot is that one of the Borg, the *Assimilator*, is going to Earth to begin the assimilation process and he needs some assistance from the collective. He contacts the *Disseminator*, who, is an editor at the Borg News Network (BNN), to see what is known about earth. The Disseminator uses the various collaborative tools and finds an old news story about the assimilation of an Earthling named Picarde who was captain of a Federation Star Ship. Picarde is now known as *Lochutus*. Disseminator finds Lochutus and they discuss the situation on Earth. Luchutus tells him about the Emperor of Earth and recommends assimilating him first. Disseminator passes the information to Assimilator and also publishes a video about the Emperor on the BNN.

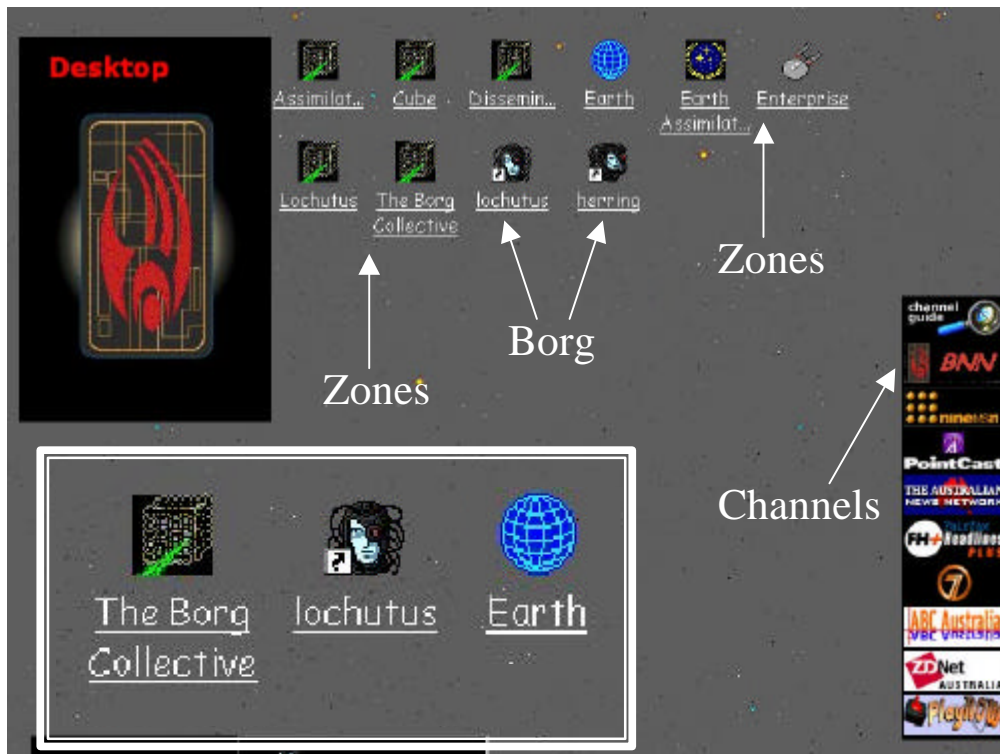


Figure 4 Desktop (inset with icons enlarged)

The opening workspace of our prototype is shown in Figure 4. This display is created through use of the Active Desktop ability to place URLs on the background. The channel bar is also placed on the background in this manner (note BNN channel). Some of the icons are shown enlarged in the lower left inset. The icons labeled Zones are file folders on a shared disk. The “Borg Collective” (an iconic representation of their cube shaped space ship) and “Earth” are zones. The “locutus” icon is a short-cut to that Borg’s local DHTML contact page.

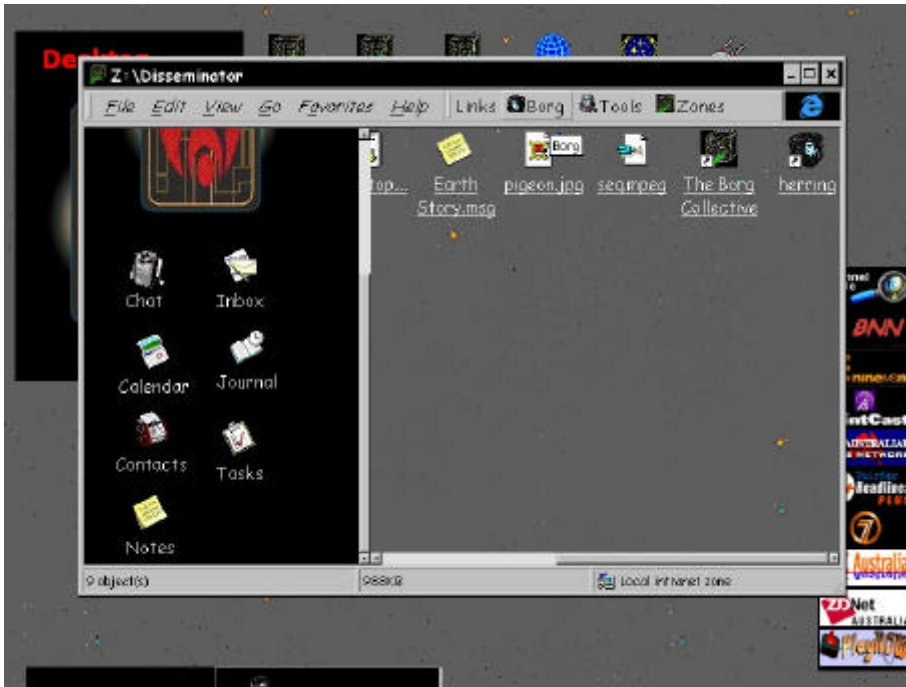


Figure 5 Browsing the Disseminator’s Zone

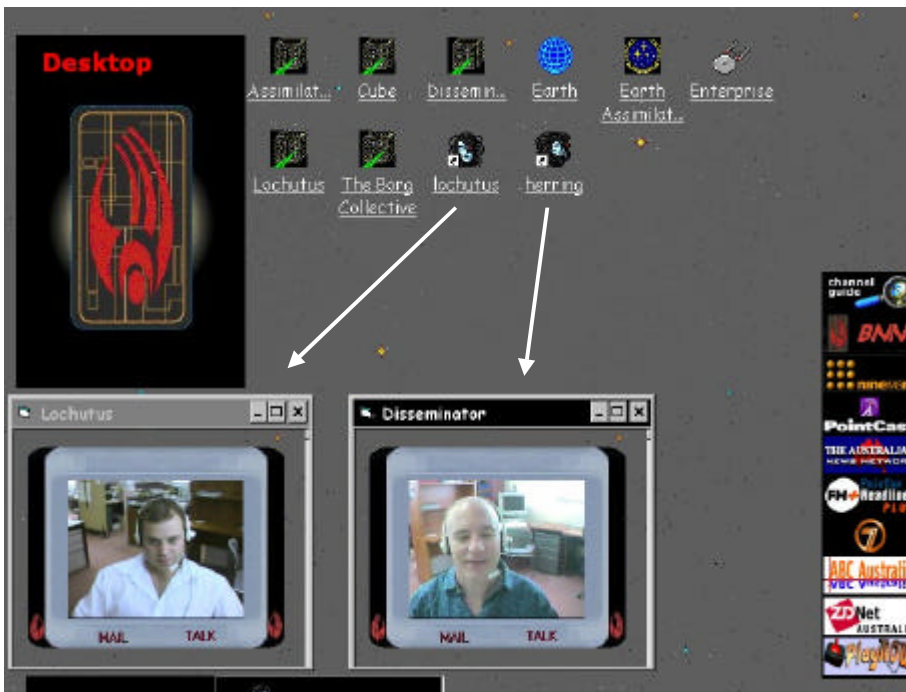


Figure 6 Video conferencing

Figure 5 shows how Disseminator's zone looks and illustrates several features. Recalling that a zone is a folder, the folder is being displayed within the browser (IE4) using a DHTML page customized for it. The frame on the right shows the contents of the zone: multimedia files, short-cuts to other zones, Borg currently in the zone, etc. The left frame shows links specific to this zone and provides a simple way to organize access to these types of facilities. Many of these links will take the user to the Exchange folders setup for this zone such as calendar, tasks, and email. However, other links can be added as needed such as the one for chat. The area of the top menu bar (pointed to by the arrow labeled utilities) is used to organize access to any common tools or information needed by all zones. For example, the Borg locator and a folder containing short-cuts to all other zones. Disseminator uses the journal option to find the story about Picarde's assimilation to become Lochutus. He then uses the Borg locator to find Locutus who is currently in the root directory of the system. Disseminator is shown in Figure 6 having established an audio/video connection with Lochutus by clicking on his icon.

Unfortunately, space prohibits showing more of the running system. But we can tell you the ending to the story. It turns out that Bill Gates is the emperor of earth, having been kept alive for centuries, and he is finally assimilated himself. This makes "Headline News" on the BNN channel. (See <http://www.dstc.edu.au/TU/staff/herring> for more screen dumps of the running system.)

6 CONCLUSIONS

We close with some reflections on our experiment and give some thought as to how CSCW systems might be implemented in the near future using the Microsoft platform. The aim of the exercise was to become "assimilated" into the Microsoft way of system development, to determine how flexible their environment is and to develop some understanding of where they are going. We spent a month exploring the plethora of software they are producing. We looked high and low: from the details of how COM works, through many software development tool kits, and to the higher level packages. We stayed on the "bleeding edge" always downloading and installing the most recent beta release. IE4 was in beta release as was Exchange 5.5. However, once we became facile with the basic technologies, it took less than two weeks to build the prototype.

A number of general lessons can be drawn from our experiences. First, there is no substitute for actually implementing something with the technology. The use of focused technology experiments seems essential if organizations are to keep up with the explosion in information technology – and not just Microsoft. The drivers of this are the rate at which software is spewing out of companies in the form of beta releases and our desire to have the latest, promised goodies. The idea that software is ever finished went away a decade ago and the idea that it will ever be completely stable has gone away now. For us this meant constantly installing and reinstalling software, corrupting parts of the system (especially the Registry), never knowing what releases would be compatible, and occasionally crashing the machine. (NB: When in doubt, reboot.) Many of these problems were due to not having any documentation or not taking time to read it if we did. To summarize, it took longer and was more frustrating than we thought it would be, but we gained some insights into how the Microsoft platform works and where it is going. We now make some specific observations on the Microsoft approach and CSCW.

Microsoft has evolved a viable component-based architecture coupled with an (almost) visual programming language. For example, Visual Basic Enterprise Edition contains a Wizard that generates an IE4-like browser from system components. We gained some appreciation for these tools in the development of our prototype. The Windows-browser integration, whether a diabolical plot or an obvious next step, is very flexible and is the basis for our system. Microsoft is moving toward a very dynamic environment based on components glued together with various scripting languages.

A good illustration of this architecture is NetMeeting 2.0 SDK. It was developed primarily as an example of how to build component-based collaborative systems and to provide second party developers with an extensible framework. It shows how various levels of components are used to customize applications. This ranges from ActiveX controls in DHTML pages, to Visual Basic and C++ programs. In many ways what we did was to turn NetMeeting "inside-out", that is, to embed the audio, video, and application sharing components into the Windows-browser architecture. The same is true of our implementation of presence

and awareness. We wanted to treat the user as a first class component of the system. The goal here is to achieve consistency of interaction at all levels. The use of short-cuts made this quite simple in the shared folder environment. Similarly, our integration of Exchange functionality within the “zones” is by DHTML links to Exchange folder short-cuts.

Now, an obvious limitation with our implementation is its reliance on a shared file system. This will not scale to the wide-area or generally distributed case. How can this problem be addressed? We think the answer is at the heart of NT5: the Active Directory. The Active Directory provides a model of the network and its resources, as well as integrates key protocols (DNS, LDAP, etc.) into an extensible framework for network management. The underlying storage manager is based on the Exchange data manager. The Active Directory schema contains classes for all network resources as well as higher level classes such as those found in Exchange shared folders (e.g. people, places, schedules, and tasks). We speculate that the Active Directory, and associated services such as Transaction Server and Message Queue, will be the enabling technology for achieving advanced CSCW-like functionality. This will result in a seamless integration of collaborative support within the user’s familiar work environment. There will be no need for separate, stand-alone CSCW systems that users must “go into” to work together.

Acknowledgements

The three Borg: Charles Herring, Andrew Lock, and Blaize Rhodes have gone their separate ways. Charles is a Ph.D. student searching for the architecture of cyberspace. Andrew contributed to Section 2 of this paper and is currently at the Distributed Systems Technology Center. Blaize is furiously programming Visual Basic at his company, Cyberdogs. We thank Professor Simon Kaplan for encouraging this effort and Michael Rees for comments. Finally, to everyone at DSTC that had to put up with the Borg invasion, you are a great bunch. Cheers.

References

1. Mansfield, T., *et al.* *Evolving Orbit: A progress report on building locales.* in *Proceedings of the ACM SIGGROUP Conference on Supporting Group Work (Group'97)*. 1997. Phoenix, Arizona: ACM Press.
2. Fitzpatrick, G., T. Mansfield, and S. Kaplan. *Locales Framework: Exploring Foundations for Collaboration Support.* in *Proceedings Sixth Australian Conference on Computer-Human Interaction*. 1996. Hamilton, New Zealand: IEEE Computer Society Press.
3. Wilson, P., *Computer supported cooperative work*. 1991, Oxford: Intellect Books.
4. Kaplan, S., G. Fitzpatrick, and C. Herring. *Centers of collaborative work.* in *Third International Conference on the Design of Cooperative Systems (COOP'98)*. 1998. Cannes, France.