THE UNIVERSITY OF CALGARY

Informal Awareness and Casual Interaction with the Notification Collage

by

Michael Lloyd Rounding

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

APRIL, 2004

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Informal Awareness and Casual Interaction with the Notification Collage" submitted by Michael Lloyd Rounding in partial fulfillment of the requirements for the degree of Master of Science.

[illegible]

Supervisor, Saul Greenberg
Department of Computer Science

[illegible]

Guenther Ruhe
Department of Computer Science

[illegible]

External Examiner, Larry Katz
Faculty of Kinesiology

April 19, 2004
Date

Abstract

Communities of intimate collaborators are people who know each other and have a need, reason, and desire to stay in contact with one another on a daily basis. These are groups such as families, research groups, and close circles of friends. The 'glue' that holds these groups together is casual interaction, or the lightweight and informal communication and coordination that is important in everyday activities. Casual interaction is facilitated by informal awareness, where people naturally accumulate information about what is happening in their physical surroundings without realizing it. The problem is that casual interaction breaks down over distance. In this thesis, I explore the design for a multimedia tool called the *Notification Collage.* Using an extendible set of independent multimedia elements (Media Items), I will show how it supports casual interaction in a community of intimate collaborators that contain a mixture of co-located and distance-separated people.

Publications

Materials, ideas, and figures from this thesis have appeared previously in the following publications:

Greenberg, S. and Rounding, M. (2001) **The Notification Collage: Posting Information to Public and Personal Displays.** *Proceedings of the ACM Conference on Human Factors in Computing Systems* [CHI Letters 3(1)], 515-521, ACM Press.

Rounding, M. and Greenberg, S. (2000) **Using the Notification Collage for Casual Interaction.** *ACM CSCW 2000: Position paper for the Workshop on Shared Environments to Support Face-to-Face Collaboration.* Philadelphia, Pennsylvania, USA, December.

Acknowledgements

I would not have been able to finish this work without the support and help of a great many people.

I want to thank Saul Greenberg, my advisor and idol in lifestyle for giving the opportunity to learn more about what you really should get out of a University degree. Saul provided me with an enormous opportunity to grow and improve as both a student and as a person. If I ever lost the faith, he was always one of the people there to set me straight and point me in the right direction again. Thanks for his patience and efforts in helping me finish this work. I hope I can do much more climbing and scrambling and hiking with him than before!

Michael Boyle will always hold a special place for me. He has been a solid place in the lab for many people, and has been a special help for me – both technically and personally – and that has not gone unnoticed. I was very proud to have organized the #4 jersey for him, because he deserves that and so much more. His support for this work and for me has been above the call of duty, and he can plan to always have me there if he needs anything! Mike's curiosity for everything life has to offer is rare and refreshing, and he will forever be one of my favourite people.

Of course the support of the people in the Interactions Lab has been astounding. Whether it be foosball tournaments or patiently using my sometimes buggy software, they have made the trip worthwhile. I've never had so much fun working with a group of people, and I hope and expect them to keep me permanently on the mailing list for hot wings excursions.

I also wish to thank my family for always being there. Darwin and Joan, or Mom and Dad, have supported me all through this endeavour, and I don't know if and how I can thank them enough for that! Phil has helped me through a ton of stuff with a laugh and a smile and a round of video games, and will always be my best friend. I want to thank Kathryn for being interested in this stuff and helping out with the project, and also for having pity sometimes when I’ve needed it, especially when I’ve needed food! I know I’m scatterbrained and I know I get frustrated, and I sometimes just plain lose hope, but all these people know me well enough to let me blow the steam off and then encourage me to get back at it again. Thanks so much for being there, and thanks for being so patient with me in times of need.

About Adena, I can’t say enough I guess. She has not been in my life for very long in the grand scheme of things, but I feel like she’s been along for this whole ride. She’s seen the high points and the low points of it, and has shown patience and support for me through and through. Thanks to her for rescuing me with a trip to the climbing gym or a day of skiing here and there, and I hope we can continue to rescue each other far into the future.

With too many names to mention, I can’t begin to talk about friends who have supported me, then worried about me, then just plain harassed me about finishing this! This has been a very worthwhile experience for me, and I think I am a different and more relaxed person than when I started.

Dedication

This thesis is dedicated to the memory of Lloyd Rounding, the smartest man I will ever have known. If I can live with even half the dignity and grace he managed over the course of his life, then I will have been a success. If you can measure the wealth of a man in terms of the respect and friendship he has legitimately earned, then he would have likely been the richest man alive. I am a better person to have known and learned from him, and am only sad that I never had the chance to tell him that he is my idol.

The best lessons aren't learned in school.

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

1.1 The Situation

People live in an active social environment. As ever, people are on the move and talking to each other about problems and solutions, micro-coordinating their activities, and sometimes just exchanging social banter.

More formally, *casual interaction* is defined as the lightweight and informal communication and coordination that is important in everyday activities (Whittaker, Frohlich & Daly-Jones, 1994). While interactions as lightweight as saying "hello" to a passing stranger satisfy this definition, my interests lie with casual interactions in *communities* of *intimate collaborators,* defined as groups of people that have a real need for close coordination and communication (Greenberg and Kuzuoka, 1999). These groups typically include friends, families and work collaborators. For these groups, casual interaction often results in purposeful communication as well as the strengthening of social bonds (Kraut, Fish, Root & Chalfonte, 1990).

The physical world presents a dynamic social environment in terms of casual interaction for intimate collaborators. People naturally accumulate information about what is happening in their environment without realizing it, resulting in *informal awareness* (Whittaker et al, 1994). For example, an open office door generally indicates presence and availability, while an office door only slightly ajar projects presence but also perhaps a desire for privacy (Kraut et al, 1990). Similarly, approaching footsteps, an opening elevator door, a phone ringing, or a snippet of overheard conversation are all cues indicating presence and availability. These serendipitous environmental cues come out of social context in that they are always available to be tracked, but may not necessarily be noticed. They can be ephemeral by being short-lived. They can serve as ‘background’ information because they need not be attended to consciously to be successfully tracked. Casual interaction is supported by these cues, as they provide for intimate collaborators opportunities to enter into conversation and potentially work.

Traditional awareness and casual interaction requires people to be in close proximity. We know that the likelihood of encountering awareness cues dramatically worsens as the distance between intimate collaborators increases (Kraut et al, 1990). This is problematic, for this loss of contact due to distance separation could prove damaging both to work practices and social relationships. For example, this is why old-style companies went to great effort to co-locate teams.

The nature of the workforce has changed recently, partly due to the wide-spread availability of high-speed internet access. Telecommuters can work from home or remote offices where they now collaborate across distance. Even family members living in different places now have new ways to communicate. To overcome these distance barriers, people who are separated now rely on computers and other technologies as a medium to help them stay in touch, to have formal online meetings – as explained below – and to casually interact with one another.

In particular, there are now multitudes of internet-based applications available that support casual interaction over distance. *Email* is the first highly successful tool to do this. More recently, *Instant Messaging* systems, *Multi-User Dungons* and *Media Spaces* offer better support for real-time distance-separated casual interactions; these are described in turn.

Instant messaging (IM) systems such as MSN Messenger, ICQ and AOL Instant Messenger let people keep a list of favoured contacts (their personal intimate collaborators) with whom they communicate, similar to an address book. These contacts (who each also have their own independent contact lists) communicate an estimate of one’s availability to others through a presence indicator. The indicator is typically updated automatically through a measure of idle time measured through keyboard and mouse inactivity, but it can also be set manually to different states such as ‘Online,’ ‘Away,’ or ‘Out to Lunch.’ Most of these systems support synchronous communication, where two or more (but most often just two) people can send brief text messages back and forth to one another in real time. Instant Messaging systems have become the current ‘killer application’ of the Internet, and their recent inclusion of multimedia (e.g., video and audio connections, groupware applications) points to where the technology is headed.

Multi-User Duneons (MUDs) are similar in purpose to IM, except that they are designed around the real world metaphor of how people meet within a physical space, such as a set of meeting rooms. MUDs simplistically simulate a physical space in a virtual environment, typically by providing its community with a set of ‘rooms’ (Curtis & Nichols, 1993). Depending on the complexity of the MUD, people navigate through different ‘rooms’ using either a text or graphical interface. When one enters a room, he or she ‘sees’ who else is in that room and can casually interact with those people as well as the objects occupying the environment. Unlike the directed conversations of instant messaging, conversations in a MUD are generally public, where any contribution or user action is seen by all room inhabitants.

Media Spaces also support casual interaction, but instead of text and awareness indicators, they use always-on multimedia channels – live video and audio links – to connect two or more participants. People maintain awareness of others by monitoring the transmitted video (which can be snapshot or continuous video), and can move into casual interaction simply by talking over the same link. While most media spaces are research prototypes, video and audio capabilities are now appearing in commercial products. For example, Microsoft’s NetMeeting and its newest MSN Messenger add an optional video and audio channel. On a different scale, video-connected meeting rooms and public places (e.g., coffee rooms and common kitchens) nurture the feeling of proximity in working communities separated by distance by exploiting how people naturally gather in common areas (e.g. Fish, Kraut & Chalfonte, 1990; Jancke, Venolia, Grudin, Cadiz & Gupta, 2001).

All of the above technologies offer provisions for casual interaction over distance. Underlying their success is peoples’ needs and desires for ways to easily stay in contact and interact while separated. These systems, while extremely simple, work because they make causal interaction possible. However, they do all suffer from different weaknesses. IM only offers a one-on-one interaction space that makes it hard for small group interactions to occur spontaneously. MUDs provide public and semi-private spaces for group interaction, but require people to be in these rooms for this to happen. This makes it hard for group members to find one another at appropriate times. Media Spaces provide very concrete awareness of others but also verge on providing too much information, raising privacy and distraction concerns. They also do not afford the awareness that could be gained from sharing electronic work artefacts such as documents.

Recognizing the strengths and weaknesses of these systems opens a novel design space. In this thesis, I explore the design for a multimedia tool called the *Notification Collage.* Using an extendible set of independent multimedia elements (Media Items), I will show how it supports casual interaction in a community of intimate collaborators that contain a mixture of co-located and distance-separated people.

1.2 Research Problems

While contemporary systems now make informal awareness and casual interaction possible, I feel that there are several problems with their designs when taken in context of how a community works together.

1. ***Impoverished awareness of the community.*** Present-day tools for informal awareness and casual interactions do not provide rich awareness of others within the *context* of the community. The research challenge is to design an environment that provides rich, multimedia awareness to the community.

2. ***Closed systems.*** Present-day tools are closed systems offering a restricted set of channels for informal awareness and casual interactions. It is difficult or impossible to rapidly extend them to afford new kinds of rich, multimedia-based interactions required by the community. The research challenge is to architect the system to be extended with little user impact as community's needs and desires change.

3. ***Impoverished interactions in the community.*** Present-day tools for informal awareness and casual interactions often provide just one channel for awareness and one channel for interactivity. Rich channels are available separately, but there is no one tool that brings them all together *and* situates them within the community. The research challenge is to provide diverse rich multimedia channels for casual interaction in the community.

1.3 Research Goals

In this thesis, I describe a tool and architecture I built that addresses each of the stated research problems.

1. ***Notification Collage Board.*** I build a tool that appropriately situates informal awareness cues and casual interactions of an entire community in a shared public space.

2. ***Media Item Architecture.*** I architect the Notification Collage Board so that I and other developers can quickly and simply extend it with Media Items that afford novel multimedia-based awareness and interactions.

3. ***Media Items.*** I build a stock set of Media Items that provide rich channels for informal awareness and casual interaction. These items will plug into the architecture so that they can appear on the NC Board as part of the community shared interaction space.

1.4 Definition of Success

Throughout this thesis, I use the term ‘success’ to both highlight existing systems and to evaluate my own. My definition of ‘success’ is quite pragmatic rather than academic.

For existing systems, I consider them successful if they are widely deployed and used in society or in a truly believable way by real life groups.

For my own inventions, I consider them successful in *use* if they are used consistently by a community for everyday interactions and purposes, and if they are voluntarily adopted by new members of that community over time. I also consider them successful as a *development* tool if programmers not directly involved in this research project can extend the system with reasonable effort, and can deploy these extensions with little disturbance to the end-user community.

1.5 Thesis Overview

In Chapter 2, I discuss the motivation for this project. I explore the sociology of Informal Awareness and Casual Interaction and some precedents that drive the requirements for the design of the Notification Collage (NC) and the Media Items it uses. I also highlight several architectural ideas that could lend their usefulness to the design of the NC.

In Chapter 3, I tell the story of the first version of the NC. I discuss its display metaphor, its placement in a research community, and how our experiences with it led to a list of re-design requirements.

In Chapters 4 and 5 I detail the redesigned NC. Specifically, Chapter 4 covers how the NC Board interface was rebuilt as a singular user interface. This includes a stock set of Media Items that were developed to enrich the community space of the NC and facilitate informal awareness and casual interaction. Chapter 5 describes how the NC's architecture was redesigned with a new client / server shared dictionary system and development model to facilitate the building of the new NC board interface. The development model is described in detail and several examples of its use when distributed to a group of average programmers are shown.

I conclude this thesis in Chapter 6 by revisiting the initial research problems and goals and summarizing their achievement. I also discuss both future directions and parallel streams for this research to highlight potential areas for improvement and further exploration in this area.

Chapter 2. Background and Motivation

As we will see in later chapters, the Notification Collage (NC) is a groupware system intended to support informal awareness and casual interaction in a group of partially co-located and partially distance-separated intimate collaborators. In this chapter, the reader is equipped with the contextual background motivating this system, sees why the NC concept is worthwhile, and gains the knowledge to understand the NC design decisions I made. I begin by re-introducing informal awareness and casual interaction, and why they are important. This sets the stage for the next section, which discusses the different granularities of groupware tools intended to support casual interaction over distance. Specifically, I present Media Spaces, Instant Messaging, Chat Rooms and Multi-User Dungeons, and Notification Systems. Finally, I present component based design and describe the idea of notification servers and how they can be capitalized on as an infrastructure for multimedia groupware.

2.1 Interaction for Intimate Collaborators

This section presents *casual interaction* and how it is supported by *informal awareness* to become the binding mechanism for communities of *intimate collaborators*. It finishes by discussing the breakdown of casual interaction and informal awareness over *distance*.

As first mentioned in Chapter 1, *intimate collaborators* are defined as groups of people that have a real need or desire for close coordination and communication (Greenberg and Kuzuoka, 1999). Intimate collaborators can include such diverse groups as families, friends, research groups, corporate teams, and so on. While the type of group can be diverse, their functions essentially remain the same: members want to further the collective goal of the group and they want to maintain the group's social structure both as a whole and on a member to member basis (Kraut et al, 1990). For the purpose of this thesis, these groups are referred to as *communities* of intimate collaborators.

2.1.1 Casual Interaction

While many communities organize scheduled meetings with the purpose of furthering their goals, there are far more *casual interactions* that occur in the workplace. These interactions are typically opportunistic or one-person initiated. For example, they occur when one person passes another in the hallway and starts an impromptu conversation, or when one seeks out another without the other's prior knowledge. Unplanned and typically brief, casual interaction (also known as informal communication) exists outside formal scheduling channels and serves to further the goals of those participating in them (Kraut et al, 1990; Whittaker et al, 1994):

- *Social greeting, pleasantries and bantering* helps maintain the social structure of the group and helps people keep up to date by informing them of 'what is going on' in another's life. This furthers social understanding in the community and how people perform group maintenance.
- *Micro-coordination* helps coordinate on-going joint activities 'in the small'. Examples include phrases such as 'let's meet in 5 minutes' or 'I've completed that thing you asked me to do' or 'I've revised the document – I'll email it so you can go through it'.
- *Seeking expertise* occurs when a person seeks out a community member who can help them. This may be by serendipitously finding the first person of many who can provide assistance, or by explicitly asking others – through casual conversation – if they can identify a person who has that knowledge.
- *Detailed discussions* occur when people move into purposeful work, perhaps by first affirming that they have the time and motivation to do so.
- *Working over artefacts* happens as people use resources ready to hand to assist their interactions – napkins, whiteboards, writing implements, scraps of paper, and documents and other items relevant to the task that happen to be close by.

All the above examples illustrate that casual interactions can easily evolve from pleasantry to more meaningful and rich work interactions.

People move in and out of casual interaction fluidly. Hallway conversations can happen in a few seconds, with barely a pause as people pass by each other. In shared work areas, people ask brief questions of others near them rather than referring to a manual. People join others at coffee tables. Even when a person is hard at work at their desk, others often find suitable times to briefly interrupt them and converse. People may pause on their way to some other destination to talk to someone who they notice. All of these conversations are unscheduled, dynamic, fluid and lightweight, yet work is being accomplished in the community (Kraut et al, 1990).

2.1.2 Informal Awareness

Informal awareness makes casual interaction possible. It is the information that people subconsciously collect about their changing environment as they move throughout their day (Greenberg, 1996). Cues as simple as hearing doors open or close, soft voices down the hall, a stack of papers to be graded on a desk or noticing someone as they walk by are all good examples of how informal awareness information is produced and collected in the physical world. It is by these mechanisms that people keep active stock of their environment and are presented with opportunities to enter into casual interaction. For example, seeing someone walk by in the hallway can both jog someone's memory about something they wanted to discuss with that person, and provide the opportunity to enter that discussion at the same time (Kraut et al, 1990).

What makes these awareness cues so successful in the real world is that they are natural and easy to act on. People do not have to actively attend to them to track them. Awareness information is more often than not gathered subconsciously in the 'background', and may or may not result in action. Cues are often ignored and missed, yet the ramifications are slight within a social context. This is a benefit, for the intense amount of information in the everyday world would quickly overwhelm us if we tried to consciously manage every bit of information. People are well-practiced at selectively attending to cues, and using these cues to help them decide whether they need to gather more information and whether they should act upon this information. The sum of the awareness cues accrued by an individual in a community allows them to form a picture of ‘what is going on’ and to make more informed decisions about contacting others and in entering casual interactions.

2.1.3 The Distance Problem

The informal awareness that supports casual interaction suffers with *distance.* While everyday cues described in the previous section are highly salient when people are in close proximity e.g., several meters, they are dramatically less salient as people are separated by even small distances, e.g., opposite ends of a corridor. The problem increases further when people are separated by greater distances, and Kraut, Egido and Galegher (1988) have shown that causal interactions decrease exponentially with distance. For example, people who are separated on different floors are only slightly more likely to come into contact and collaborate than those who are separated in different buildings or cities (Kraut et al, 1988). The lack of co-presence leads to a lack of opportunity to interact.

The decrease in visibility that occurs over even small distances also hurts the richness of messages being sent and received between collaborators. They can no longer send auxiliary messages such as facial expressions or nodding accompanying understanding. Also, they can no longer directly hear one another, increasing the cost and reducing the spontaneity of any interaction. The lack of these mechanisms can lead to a breakdown in grounding, where it is more difficult for people to gain a common understanding of their respective environments (Clark & Brennan, 1991).

In the distant past, distance separation was not a huge problem. Extended families (where all family members lived together or close by) were common. Social communities were small (e.g., villages). Businesses were small and tended to gather all people at one site. This has, of course, changed in our much more global community. In contrast, nuclear families are the norm in Western society. Cities sprawl very large areas. Even modest-sized businesses have workers scattered over different floors of a building or across buildings. International companies almost always have offices in different cities. In spite of distance separation, many of the people that comprise these communities still need to remain intimate collaborators. This is where computers can help, particularly by providing tools that support informal awareness and causal interaction spanning the distance barrier.

2.2 Tools for Casual Interaction

The previous section highlighted the importance of casual interaction supported by informal awareness in improving group dynamics and getting work done (Kraut et al, 1988). However, the distance problem is now becoming more prevalent as people tele-commute or families move to different cities. There is an understood need for the grounding and communication that casual interaction provides (Clark & Brennan, 1991), and thus also a desire to produce tools that solve the distance problem and allow people to casually interact over distance. This section will provide an overview of tools available to people and how they are intended to be used. They range in audience size anywhere from one on one chatting to shared video feeds to full public displays connecting public spaces and chat rooms. As we will see, systems vary in the group size (e.g., from two to thousands) and the private versus public nature of the interactions (e.g., from private chat windows to public displays).

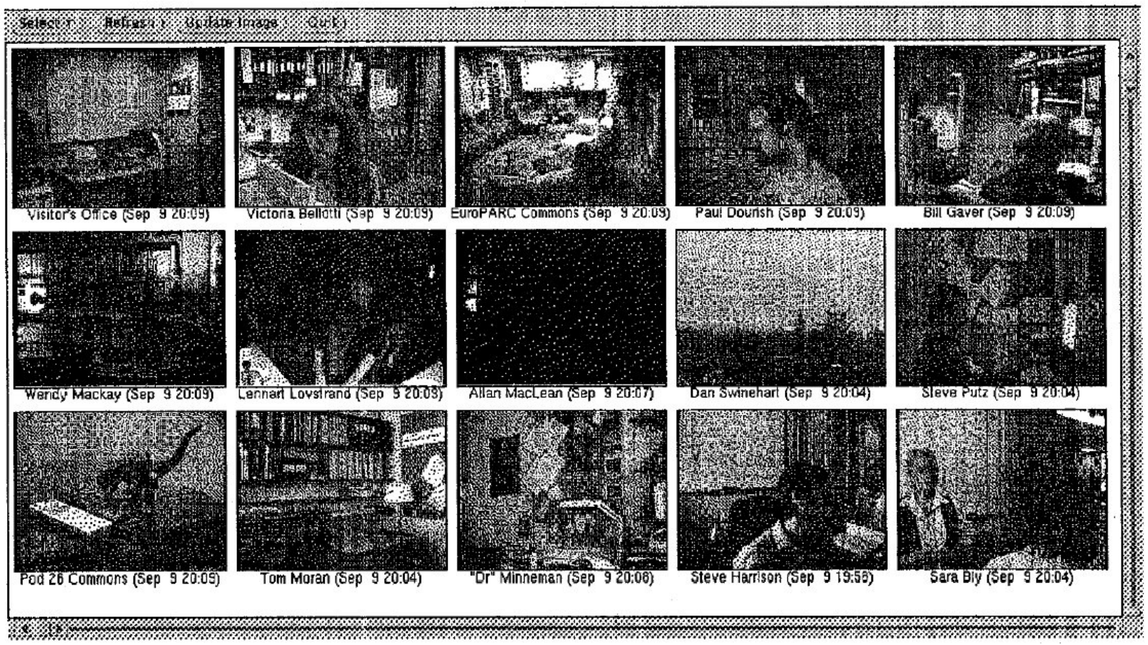
2.2.1 Media Spaces

Traditionally, media spaces are a class of applications that use ‘always on’ rich media channels such as audio and video to allow remote collaborators to be aware of one another’s activities and to easily move into dyadic conversations. However, today media spaces come in many forms: from simple one on one desktop systems, to video walls connecting different common areas. They all share the common attribute of using ‘always on’ video as a means of determining availability and communicating with others (Bly, Harrison & Irwin, 1993).

Systems such as **CAVECAT** provide full audio and video connections between desktop locations (Mantei, Baecker, Sellen, Buxton, Milligan & Wellman, 1991). A CAVECAT setup consisted of a video camera, a TV monitor, a personal computer and a microphone. Through the system people could participate in up to four point meetings, with the TV monitor showing all four points in quarters. The system was used in faculty offices and a group workspace to connect people in a co-located setting. It was used both to have formal meetings online and also to host brief encounters between sites (such as introducing someone to the group through the system and not actually leaving an office). While the system was used successfully to perform the described activities, it also suffered from poor audio filtering that became distracting and detracted from the user experience.

Cruiser is a similarly constructed traditional media space that attempted to facilitate informal awareness and casual interaction by providing one and three second video connections into others' workspaces (Fish, Kraut, Rice & Root, 1993). The one second *glances* and three second *cruises* were initiated by users via a system command to look into another's space, while simultaneously broadcasting the initiator's video (for reciprocity). Not specifying a specific person as a target caused the system to initiate glances or cruises with a series of randomly selected people. In addition, a third opportunity for awareness was provided through *autocruises,* where the system automatically opened a cruise between two people at random times. The attempted metaphor was that of walking down a hallway and briefly glancing into others' offices. In practice, the auotocruise facility proved to be invasive and distracting, and users tended to not even use the system. While it did have a *privacy* feature, people opted instead to disconnect completely from the system. The facilities for informal awareness and casual interaction provided in the system were not as subtle as their real life counterparts.

While traditional media spaces as described above provide awareness and interaction through full audio and video connections, **Portholes** takes a low-cost approach by using simple snapshot video (Dourish & Bly, 1992). Through it, research groups within Xerox located in both Europe and in California were able to maintain contact with one another. While several clients for image display were created, the simplest ("pvc") showed an array


Select
Refresh
Update Image
Quit
Visitor's Office (Sep 9 20:09)
Victoria Bellotti (Sep 9 20:09)
EuroPARC Commons (Sep 9 20:09)
Paul Dourish (Sep 9 20:09)
Bill Gaver (Sep 9 20:09)
Wendy Mackay (Sep 9 20:09)
Lennart Lovstrand (Sep 9 20:09)
Allan MacLean (Sep 9 20:07)
Dan Swinehart (Sep 9 20:04)
Steve Putz (Sep 9 20:04)
Pod 26 Commons (Sep 9 20:05)
Tom Moran (Sep 9 20:04)
"Dr" Minneman (Sep 9 20:06)
Steve Harrison (Sep 9 19:58)
Sara Bly (Sep 9 20:04)
Figure 2.1 – the Portholes ‘pvc’ client (from Dourish and Bly, 1992)

of video nodes as thumbnails (Figure 2.1). Each thumbnail had a time associated with its capture time, and could be used to email to the person represented by the video node. Another Portholes client, ‘*edison*’, allowed people to associate an audio message with their node image that others could retrieve. The ’*viewmaster*’ client was the public client, and loading it would only allow the user to display public location nodes; no personal office nodes were included. This simple system successfully fostered a feeling of community for the two sites, and elicited comments such as “I like to see the day start in America when people are arriving” or “It’s fun to see the sun rise in the UK”. One of the chief complaints of the system was that when someone wanted to run it, it required a significant amount of screen space to display it.

On a different scale, **VideoWindow** tries to connect two public common areas transparently (Fish, Kraut and Chalfonte, 1990). It does so by projecting a very large video image (eight feet by three feet) of the other location onto one wall. Microphones and speakers are set up strategically so that standing and talking at a particular spot in one location will sound like speech is properly originating from the projected image in the other location. The idea was that people would be able to notice one another and converse over the link just as if they were conversing with someone in the same room, with the same feeling. An informal study by Fish et al (1990) found that people took advantages of virtual opportunities to casually interact across VideoWindow. They also found that people *more frequently* took advantage of co-located opportunities to casually interact. E.g., in the same time-frame that VideoWindow was being observed, a higher proportion of the opportunities occurring in one room were capitalized on than those occurring over VideoWindow.

The **Video Kitchen** project at Microsoft Research also takes the approach of linking public spaces – common kitchens – between different floors and buildings (Jancke, Venolia, Grudin, Cadiz & Gupta, 2001). Three kitchens were outfitted with microphones, cameras and a video projector. The projector displayed three kitchens in each of three quarters of the display, while the fourth quarter was reserved for CNN, in an attempt to provide something to draw people into conversation through the link. While initially setting up the system, the researchers ran into resistance from people not wanting to be on camera every time they walk into the kitchen. To resolve this issue, an 'off' button was placed at the door of the kitchen. If pressed as one enters the room, the feed from the room's camera would turn off for 180 seconds. In practise, people actually did use Video Kitchen to foster casual interaction. However, acceptance was far from universal and there still were those who would hit the 'off' button on entry, even when other people in the kitchen were conversing through the Video Kitchen link.

Media spaces have been likely the most developed type of groupware application aside from shared drawing canvases. Yet they remain mostly research systems and have not entered real mainstream use. Still they teach us lessons we can apply to new designs.

a) The rapid ability to gauge awareness and move into conversation over a video and audio channel supports the intermittent, spontaneous and rapid nature of casual interaction.

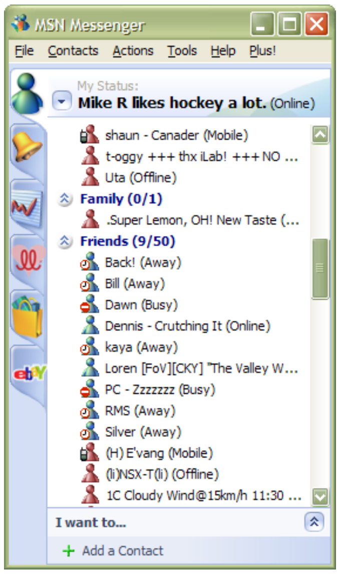
b) Video as a medium is very successful in terms of judging activity and availability of a potential contact.

c) From the many field deployments of media spaces, we know that privacy is a problem. At the very least, participation in a collaborative space should be voluntary, i.e., people should not be forced to participate simply by walking into a room that contains a media space.

2.2.2 Instant Messaging

Instant messaging (IM) is a class of conceptually simple groupware applications that offer mostly text-based impromptu real-time communication and awareness via the internet. IM systems such as ICQ, AOL Instant Messenger and MSN Messenger are simple in concept but are now claimed to be used by millions (Nardi, Whittaker & Bradner, 2000), highlighting that tools such as these are considered important for supporting informal awareness and casual interaction with our collaborators.

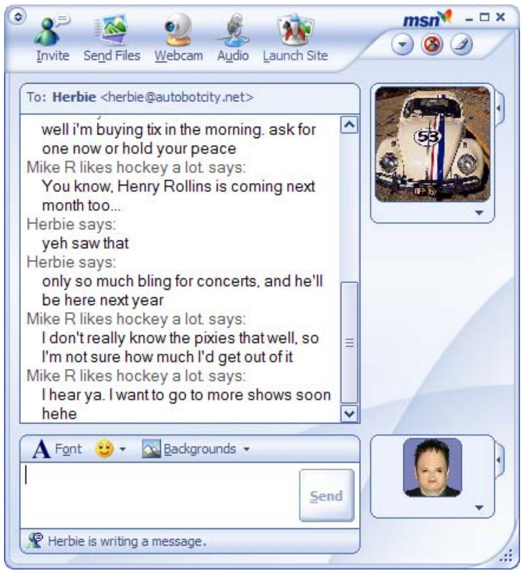
IM systems allow people to track a list of 'buddies,' or contacts they communicate with (MSN Messenger's list is shown in Figure 2.2a). People can communicate their availability to others using iconic status indicators representing different states such as 'online', 'away' or 'busy' (as illustrated on the left side of Figure 2.2a). Leaving the computer for a certain amount of time will also automatically adjust the online state to



Mike R likes hockey a lot. (Online)

shaun - Canader (Mobile)
t-oggy +++ thx iLab! +++ NO ...
Uta (Offline)
Family (0/1)
.Super Lemon, OH! New Taste (...
Friends (9/50)
Back! (Away)
Bill (Away)
Dawn (Busy)
Dennis - Crutching It (Online)
kaya (Away)
Loren [FoV][CKY] "The Valley W...
PC - Zzzzzzz (Busy)
RMS (Away)
Silver (Away)
(H) E'vang (Mobile)
(li)NSX-T(li) (Offline)
1C Cloudy Wind@15km/h 11:30 ...

Figure 2.2a – MSN Messenger's Buddy List



To: Herbie <herbie@autobotcity.net>

well i'm buying tix in the morning. ask for one now or hold your peace
Mike R likes hockey a lot. says:
You know, Henry Rollins is coming next month too...
Herbie says:
yeh saw that
Herbie says:
only so much bling for concerts, and he'll be here next year
Mike R likes hockey a lot. says:
I don't really know the pixies that well, so I'm not sure how much I'd get out of it
Mike R likes hockey a lot. says:
I hear ya. I want to go to more shows soon hehe

Herbie is writing a message.

Figure 2.2b – MSN Messenger Chat window

'away' or some equivalent. Double clicking a name in the contact list invokes a chat window where a one on one chat can occur (Figure 2.2b). Nardi, Whittaker and Bradner (2000) found that people saw the ability to quickly get a feeling for 'who was around' in their personal contact list was important, as well as the lightweight interface for entering into a conversation.

Although the contact list can be perceived as a personal 'group', IM's awareness and resulting conversations are typically dyadic. This makes sense for the context of IM: each person has differing contact lists, with potentially many personal social and work circles being represented (e.g., the 'family' and 'friends' classifications in Figure 2.2a). That is, 'groups' are not in common between IM holders. Although close individuals may have overlapping people in their contact lists, they will generally be different.

IM systems also do not broadcast conversations beyond those (usually two people) involved in them. The consequence is that people remain unaware of the conversational state of the people on their contact lists. Compared to real life, we may know someone is there, but we cannot see if they are talking to others or hear what they are talking about.

IM's availability state, while simple, leads to inaccuracies. For example, a person who appears 'Online' may actually miss invitations to converse simply because they are not looking at the screen, or have just stepped out, or are engaged in other activities. This can actually serve as a resource allowing people to mitigate privacy through 'plausible deniability' (Nardi et al, 2000). A lack of response to an IM message is not seen as rude; it is perfectly acceptable if a person whose status says 'online' is not actually there. For this reason, it has also been observed that many people begin IM conversations with questions such as 'are you there?' As well, some explicitly set their availability to 'away' or some equivalent making it less likely that they will be interrupted by others. They essentially hide while online.

IM conversations are low-cost. They can be ignored for later response or they can be used for a quick question, which easier than a phone call or walking down a hallway to find someone. More recently, IM has begun to offer bridges into richer interaction channels such as video, audio and shared applications (e.g., the Webcam, Audio and Launch Site buttons in Figure 2.2b). Also, additional people can be added to conversations to increase the circle of a discussion (e.g., the Invite button in Figure 2.2b).

The rapid success of IM highlights the importance of the lightweight awareness they provide. While they are not tailored to communities of intimate collaborators, we can take some lessons from their successes.

a) Even minimal awareness of the presence of intimate collaborators is seen as important in gaining a feeling of 'who is around'.

b) Establishing conversations should be lightweight, where it is easy to connect and converse with others.

c) Interactions can be ignored and left in the periphery until such time as it is seen as convenient to respond.

2.2.3 Chat Rooms and MUDs

Chat rooms allow people (often strangers who may not know one another) to enter and chat in a large, public arena. Unlike IM, a chat room defines a public group space, where all see who is in the room and can overhear and participate in all conversations.

Chat rooms are popular. Systems such as Internet Relay Chat (IRC) see daily use of thousands of users. More recently, web-based services such as MSN or Yahoo! and online video game services such as Battle.net have also begun offering live chat rooms as part of their sites.

The 'group' and conversation defined by a chat room is often anonymous and in a state of flux. While regular users may appear daily in particular chat rooms and get to know one another through their online presence, new people are also continually arriving and departing. Users arriving also have no pre-existing knowledge of what is happening in a particular chat room. The only clues or motivation people may have to join a chat room is its name, which is often used to indicate an intended discussion topic. In reality, the conversations actually taking place can diverge significantly from these topics. Still, chat rooms are successful in that they allow people to interact and get some work done. For example, Blizzard Entertainment's Battle.net chat rooms are extremely successful in allowing anonymous users from diverse locations to negotiate and enter both competitive and cooperative gaming sessions.

Multi-user Dungeons (MUDs) use a richer version of the room-based metaphor for moving about a virtual space that contains objects as well as other people (Curtis & Nichols, 1994). Traditional MUDs have come in the form of online text-based role playing games. An online database contains the definition of the 'world' to which people connect. The user signs on and takes control of an avatar, which they use to move through different 'rooms' in the world using text based commands such as "`go west`". Objects can also be stored in the database with which users can interact in the environment. For example, they could use commands like "`read sign`" or "`climb ladder`" or "`open door`". The worlds are extensible from within, allowing users with correct access to add, modify and remove rooms and objects, as well as changing the way in which the world works. All of this is done with a native programming language.

Seeing potential for supporting community communication in the room-based metaphor of the MUD model, Xerox PARC developed a customized server called **LambdaMOO** which allowed them to add a graphical interface (Curtis & Nichols, 1994). While traditional MUDs incorporated only text based interactions through a standard telnet client, the LambdaMOO server could send messages to its specialized clients that could invoke windows to provide a different interface. The client tracked interaction on these windows and sent any relevant events (e.g., mouse button clicks) back to the server for processing.

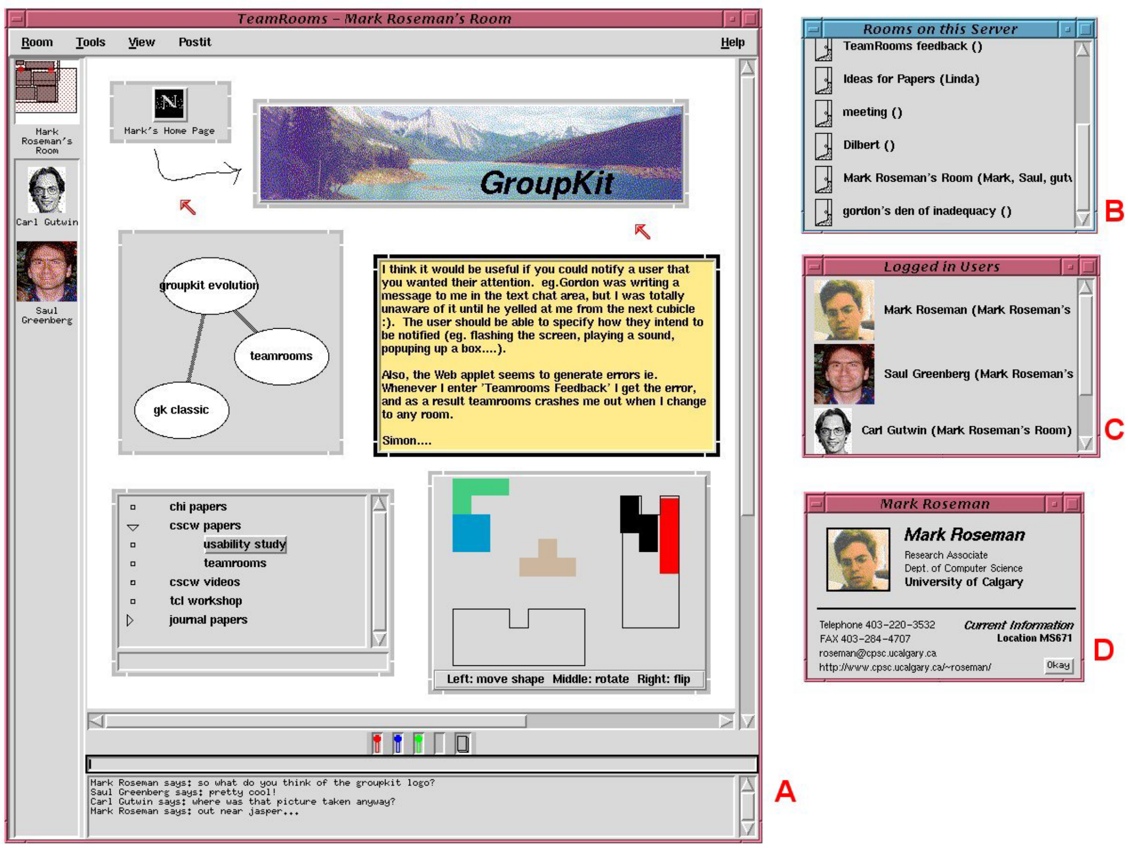
The benefit of the graphical interface elements is that it opened the possibility of adding rich media such as video and images to windows for people to share and converse over. Later versions included an audio channel for each 'room' where audio sourced from each person occupying the room was heard. As people navigated through the rooms, the server would process the channels and have the clients multicast the appropriate audio between one another. Because processing of the channels themselves was performed on the server, it could also 'mix in' a smaller volume audio from adjacent rooms, somewhat akin to overhearing a conversation through a doorway (Curtis & Nichols, 1994). While the video was very low fidelity (320 by 240 greyscale at five frames per second), it was actually enough that people could tell ‘what was going on’ in another space, allowing them to judge the availability of other people.

The LambdaMOO MUD concept was taken another step towards support for communities with the **AstroVR** system, developed by Xerox PARC and the NASA/JPL Infrared Processing & Analysis center (Curtis & Nichols, 1994). AstroVR is a modified version of LambdaMOO with the purpose of supporting researchers – rather than a broad community of strangers – collaborating on astronomical research. The community already shared much of their research over the internet using email, the web, and USENET newsgroups. People from outside the community (the public at large) added their own questions and comments to the community newsgroups, and as a result the levels of discussion also had to match the level of the public, reducing the effectiveness of the community’s online collaboration. AstroVR provided researchers with an exclusive online forum for collaboration within their community. It included tools for sharing images of research via links, ‘meeting rooms’ where one person could direct an annotated slide show and record discussions about it, as well as internal email and bulletin board systems – these messages never left the confines of the MUD system.

While the progression of these systems succeeded in narrowing down the communication circle to groups such as those communities described in section 2.1, they still had the usage hurdle of being primarily text-based systems at heart. **TeamRooms** (Roseman & Greenberg, 1996) takes on the room metaphor of the MUD, but also moves completely away from the text-based interfaces (Figure 2.3). TeamRooms was developed using GroupKit, a Tcl based toolkit for building groupware (Roseman & Greenberg, 1996). It provided a community with a set of virtual ‘rooms’ where they could create graphical user interface (GUI) objects and interact over them. GUI objects persisted so that things in a room were there (and in the same state) when people returned. Also, all objects are groupware, so the act of entering a room puts people in immediate contact with one another and with the room’s contents. A list of the rooms available was provided on a small dialog

(Figure 2.3b). Objects were small applets such as brainstorming tools, or pictures, or postit notes (the objects on the white canvas in Figure 2.3a), and were added via a menu. Each room also had a standard set of tools, including a chat system (bottom of Figure 2.3a) and its wall was an electronic whiteboard for sketching and annotation (the white canvas in the background of Figure 2.3a). TeamRooms also included a way for third party developers to create and add their own custom applets.

A list of people who were in a room was shown on the left-hand side of the canvas, providing some sense of who was around (Figure 2.3b). Once in room, people could interact in real time and gesture via tele-pointers and drawing. To provide even more community awareness, a list of people currently using the system was available on another dialog, showing also which room they were in (Figure 2.3c). As well, contact information for particular users was available (Figure 2.3d).



I think it would be useful if you could notify a user that you wanted their attention. eg.Gordon was writing a message to me in the text chat area, but I was totally unaware of it until he yelled at me from the next cubicle :). The user should be able to specify how they intend to be notified (eg. flashing the screen, playing a sound, popuping up a box....).

Also, the Web applet seems to generate errors ie. Whenever I enter 'Teamrooms Feedback' I get the error, and as a result teamrooms crashes me out when I change to any room.

Simon....

```
Mark Roseman says: so what do you think of the groupkit logo?
Saul Greenberg says: pretty cool!
Carl Gutwin says: where was that picture taken anyway?
Mark Roseman says: out near jasper...
```

TeamRooms feedback ()
Ideas for Papers (Linda)
meeting ()
Dilbert ()
Mark Roseman's Room (Mark, Saul, gutv
gordon's den of inadequacy ()

Mark Roseman (Mark Roseman's
Saul Greenberg (Mark Roseman's
Carl Gutwin (Mark Roseman's Room)

Mark Roseman
Research Associate
Dept. of Computer Science
University of Calgary

Telephone 403-220-3532
FAX 403-284-4707
roseman@cpsc.ucalgary.ca
http://www.cpsc.ucalgary.ca/~roseman/
Current Information
Location MS671

Figure 2.3 - the TeamRooms Interface (From Roseman & Greenberg, 1996)

While the system had extensive collaborative facilities, in actual practise it was found to be too heavyweight for community awareness over distance (Greenberg & Kuzuoka, 1999). The main problem was that people had to be actively in the system and looking at it to gain awareness. While successful in enabling interaction, moving through the startup and login process was too effortful if one only wanted to gather awareness information (e.g., if anyone was present). As well, the size of the application was seen as too costly to keep on the screen for the sole purpose of gathering awareness cues from the community.

In summary, chat rooms evolved into MUDs, then multimedia MUDs such as LambdaMOO and AstroVR, and ended up with TeamRooms, a MUD-like system enabling collaboration over persistent groupware applets. The progression from purely text based systems to graphical systems allowed developers to add more novel multimedia information to their spaces. I take several lessons from these systems.

a) Rooms are a good metaphor for grounding group interaction and are well accepted in large group systems, but a single room could suffice for a simple community system wanting group awareness.

b) People benefit from interacting over objects and work artefacts as they can encourage interaction more than simple text-based chatting.

c) Systems must be lightweight to get started and use.

d) Large amounts of multimedia information end up taking up a lot of space, and are thus too costly to keep on the primary screen. Instead, perhaps ephemeral awareness information could be moved off the primary display to secondary displays, or even to large public displays (Grudin, 2001).

2.2.4 Notification Systems

Notification systems simply display information as it comes in. That is, people publish information to the system, and subscribers see the information as it is propagated. They have enjoyed considerable use and attention from a large community of people using them (Fitzpatrick, Kaplan, Mansfield, Arnold & Segall, 2002).

Coffeebif is a simple example of such an application (Fitzpatrick et al, 2002). It was developed for an organization with a single coffee room to foster how people – who were distributed over several floors – met over coffee. It is a small window with a picture of a coffee cup. Anyone running Coffeebif is connected to a common 'place' where Coffeebif information is sent. When a person gets up to go to the coffee room, they simply click on the coffee cup to send a message, which increases the count of people on a coffee break and adds the new person's name to a scrolling list of people in the coffee room. The information is broadcast to all others.

TickerTape is another application running on the same system as Coffeebif that simply presents a one line scrolling marquee of notifications (Fitzpatrick et al, 2002). Via a simple interface, people can choose any number of 'groups' from which they wish to receive notification (such as the 'coffee' group accompanying Coffeebif). People can then post messages to these groups which are automatically sent to all subscribers (for example, when the number of people out for coffee in Coffeebif reaches five, TickerTape notifies people subscribing to the 'coffee' group that a large group has gathered). Incoming messages steadily scroll across a one line interface. Messages are textual but can also contain MIME attachments such as URLs or pictures. For attachments, small icons accompanying the scrolling notification can be clicked to retrieve them. Messages will fade over time, eventually disappearing from the message queue altogether.

TickerChat runs in the same venue as TickerTape, but was borne from the desire to more easily move into discussions over the same medium. Instead of an ephemeral scrolling list of messages, it takes on the form of a traditional chatting interface, logging notifications as a list as they arrive in time order. This allows a more persistent state to the information as it arrives, and makes it easier to review and discuss.

We can take a major lesson from these simple, yet elegant systems.

a) The ability to 'broadcast' information of any kind to interested parties is important, for it serves as a means to supply awareness, to provide information, and to 'dig deeper' into that information if desired (i.e., through message attachments)

2.3 Technical Resources for Design

While the previous section considered what groups see and use, this section concentrates on infrastructure. I present two concepts that are useful for developing multimedia groupware. First, *Notification Servers* can propagate data as events from one client to others, who can then react to the information. Second, *Component-based Design* allows the re-use of binary code and objects that can be loaded on the fly. As described in a later section, both of these ideas are used in the Notification Collage to design a system that is constructed as a modular software system that allows for easy distribution, connection and expansion.

2.3.1 Notification Servers

As stated by Fitzpatrick et al (2002), "the function of a notification service is to act as a distributor for descriptions of events". That is, when some event occurs, those who are listening for it are informed of the potential change in state. This is simply called a *notification.*

Patterson, Day & Kucan (1996) described the Notification Service Transfer Protocol (**NSTP**), which was initially intended as a concept for building synchronous groupware. In it, clients requested information from a server on state changes for some set of information. Specifically, it is concerned with the concepts of Places, Things, Facades, and the server itself Figure 2.4a illustrates how these are interrelated with respect to clients waiting for notifications.

The notification server itself is connected to by clients. Multiple places can exist in the server, each representing the state of a different system or set of information. The things in the place are the objects that actually hold the state information of the system. Facades can be seen as the public face of a place. For instance, it may be the case that some clients may not need to receive notifications for all the things in a given place. By only viewing the façade, they are spared from the unnecessary notifications that would otherwise be generated. Of course, there is nothing to stop a client from gaining access to all of the notifications generated.

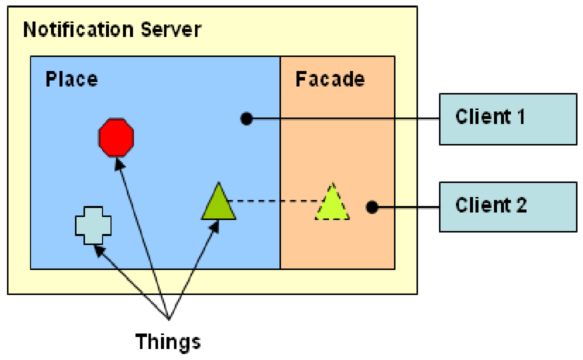


Figure 2.4a – components of NSTP (Patterson et al, 1996)

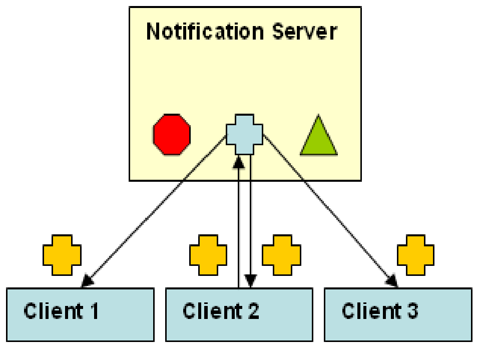


Figure 2.4b – a typical notification sequence (Patterson et al, 1996)

Notifications themselves are demonstrated in Figure 2.4b. When client 2 changes some data e.g., to change the colour of the center object, the server takes note of it and then propagates the change as an event that carries the attributes of the change out to the other clients who are occupying the place or façade, allowing them to take action on the change if necessary. It should also be noted that it is perfectly acceptable for clients to occupy more than one place at a time.

The ephemeral nature of awareness cues and information makes this model a good choice for designing awareness systems. The environments we are trying to support always entertain some current state of activity. Notification servers can be used to track what is happening in the environment by having clients 'call in' with different pieces of information to offer which when treated as a whole represents the community (Patterson et al, 1996).

A successful deployment of such a notification server system is the **Elvin** system (Fitzpatrick et al, 2002). Elvin is a 'pure' notification server (e.g., an active client generating information tells the server about it, which in turn tells other passive clients about the changed information). In particular, the three simple applications already described in section 2.2.4 – all based on Elvin – have proven enormously successful across a broad user-base.

2.3.2 Component Based Design

A big problem with building rich, multimedia based groupware systems is re-use and manageability, especially if many people are using and designing the system at one time. Having one large program doesn't make sense as it becomes impossible to manage all the separate developer versions. Still, as group dynamics change over time, so might group requirements of our proposed design. Thus, it is not an unrealistic requirement that a system designed to support a community may need extension at some point.

MUDs (Curtis & Nichols, 1993) recognize this by their ability to add and change objects and interactions in their virtual environments. This is all done in native languages within the MUDs themselves. TeamRooms makes the same effort by allowing developers to build basic applets, and simply load them in the application to make them groupware (Roseman & Greenberg, 1997). Arrangements such as these definitely hold the advantage that the systems can be extended. However they limit the interaction potential for new object design because the developer is limited to the capabilities of the native languages being used to create these objects.

Another alternative is component based design, which can take a more 'plug-in' based approach to design. Component-based design is essentially re-usable, pre-packaged program objects or functionality. The developer simply publicizes an Application Programmer's Interface (API) or contract for how their component works, and then that component can be included in any other program which can invoke the API. An example of this is ActiveX in Microsoft Windows. Many common objects in Windows are built as ActiveX objects (also called Component Object Model or COM objects), which can be inserted and used in many different applications without having to rebuild them from scratch. For example, many common widgets are implemented as ActiveX objects, e.g., the Command Buttons we see on many forms and dialog boxes are all derived from the same ActiveX control.

For groupware, extensions to group systems can also be built as new components, all employing an agreed upon contract, such that the system would always know how to load these objects. This opens the design space for novel interactions by allowing designers to package a multitude of techniques in their sharable objects, as long as they comply with the standard software contract.

Creating extensions in this fashion also contains deployment benefits. It opens the avenue for loading newly developed objects on the fly, dynamically downloading the required binary to use the new object. People using the system would never have to obtain a brand new version to take advantage extensions, nor would they have to even restart it to gain new interaction opportunities. Developing the system and extending it can remain two distinctly separate development streams only using the development contract (the API) to inter-operate.

2.4 Summary

In this chapter, I have described casual interaction and informal awareness within the context of a community of intimate collaborators. Casual interaction is supported by informal awareness cues, and it is used by community members to maintain the social makeup of the group. I also explained, however, how these mechanisms break down when physical propinquity dissolves. As casual interaction is seen as being so important, many different kinds of tools have been developed with the aim of bridging this distance gap. I have discussed tools varying from media spaces to instant messaging and MUDs, systems that range from small research community use to widespread internet use. While these systems have different intended audiences and present different granularities of awareness information, my particular interests revolve around small communities.

Based on these concerns, I built an application incorporating a single ‘room’ (MUDs and Chat Rooms) where lightweight multimedia notifications (reflecting the diverse media in Media Spaces, Instant Messaging, and Notification Systems) are displayed as discrete component objects within a MUD-like system (taking inspiration from MUDs and component based design). The system posts input gathered from remote locations to a large display in the community’s space, as well as peoples’ personal desktops. All of the communication passes through a notification server that notifies connected clients when someone posts new information and caches the current state of the community's data. The system is designed to present info as lightweight and peripheral objects. It does not demand the full attention of its users: rather it can be attended to in passing, where people collaborate should the need or desire arise. In the next chapter, I present the initial design and usage experiences with the Notification Collage.

Chapter 3. The Notification Collage

In this chapter, I present the initial vision for the Notification Collage (NC), a groupware system in which distributed and co-located colleagues comprising a *community* of *intimate collaborators* post *multimedia elements* onto a real-time collaborative surface that all members can see. Akin to collages of information found on physical bulletin boards (Kerne, 1997), the NC randomly places incoming elements onto this surface. People can post assorted media: activity indicators, slide shows displaying a series of digital photos, snapshots of a person's digital desktop, textual sticky notes, and a host of others.

I begin with a description and design rationale for the NC. This is followed by a description of our research group's early usage experiences with it, concentrating on how group members bootstrapped themselves, how NC was used in both public and personal spaces, and how people interacted on it. A set of strengths and weaknesses derived from these experiences results in a redefinition of the initial vision. It should be noted that much of the information in this chapter is taken from a published research paper about the Notification Collage (Greenberg & Rounding, 2000).

3.1 NC Design Rationale and Description

Our system design follows the metaphor of a bulletin board containing a collage of randomly positioned and possibly overlapping visual elements (Figure 3.1). The original Notification Collage is illustrated in Figure 3.2. Using various client programs (e.g., as in Figure 3.3), members can post a variety of multimedia elements (called Media Items) to the NC. Upon receipt, the NC reconstructs these as discrete visual entities and randomly places them on the left side of the vertical separator bar that splits the board. Overlapping Media Items are allowed, and new items are always positioned on top.

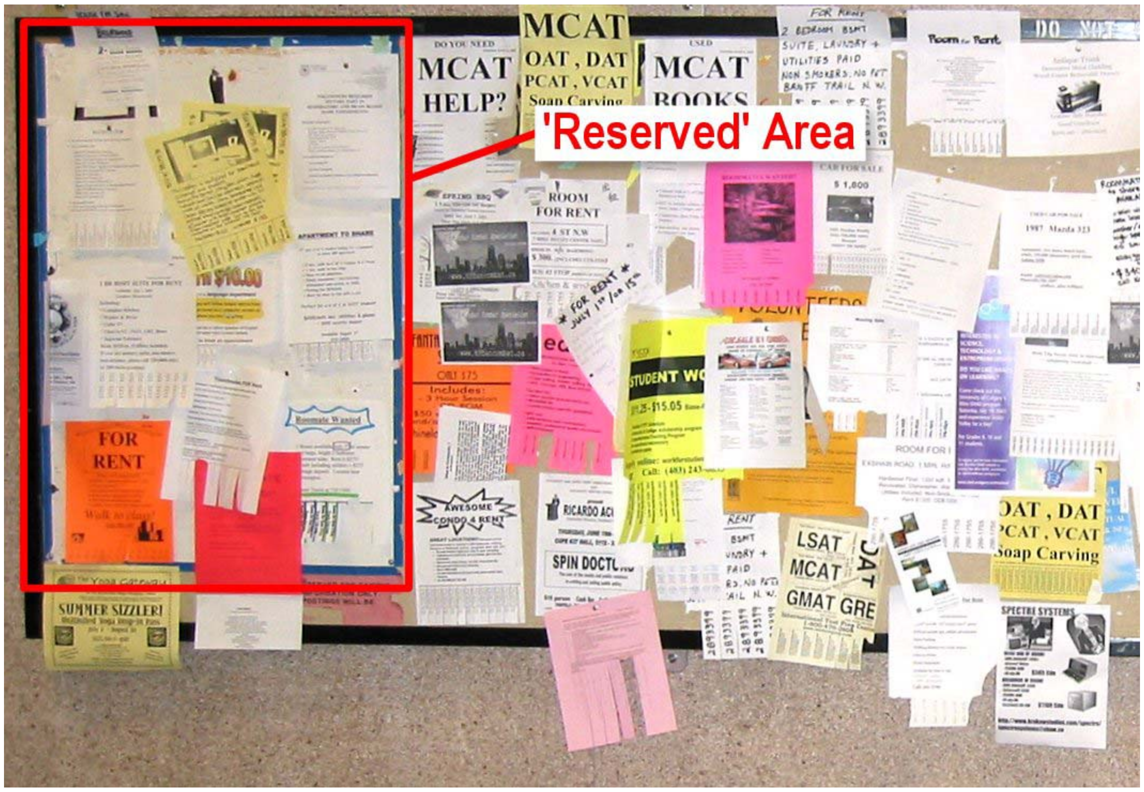
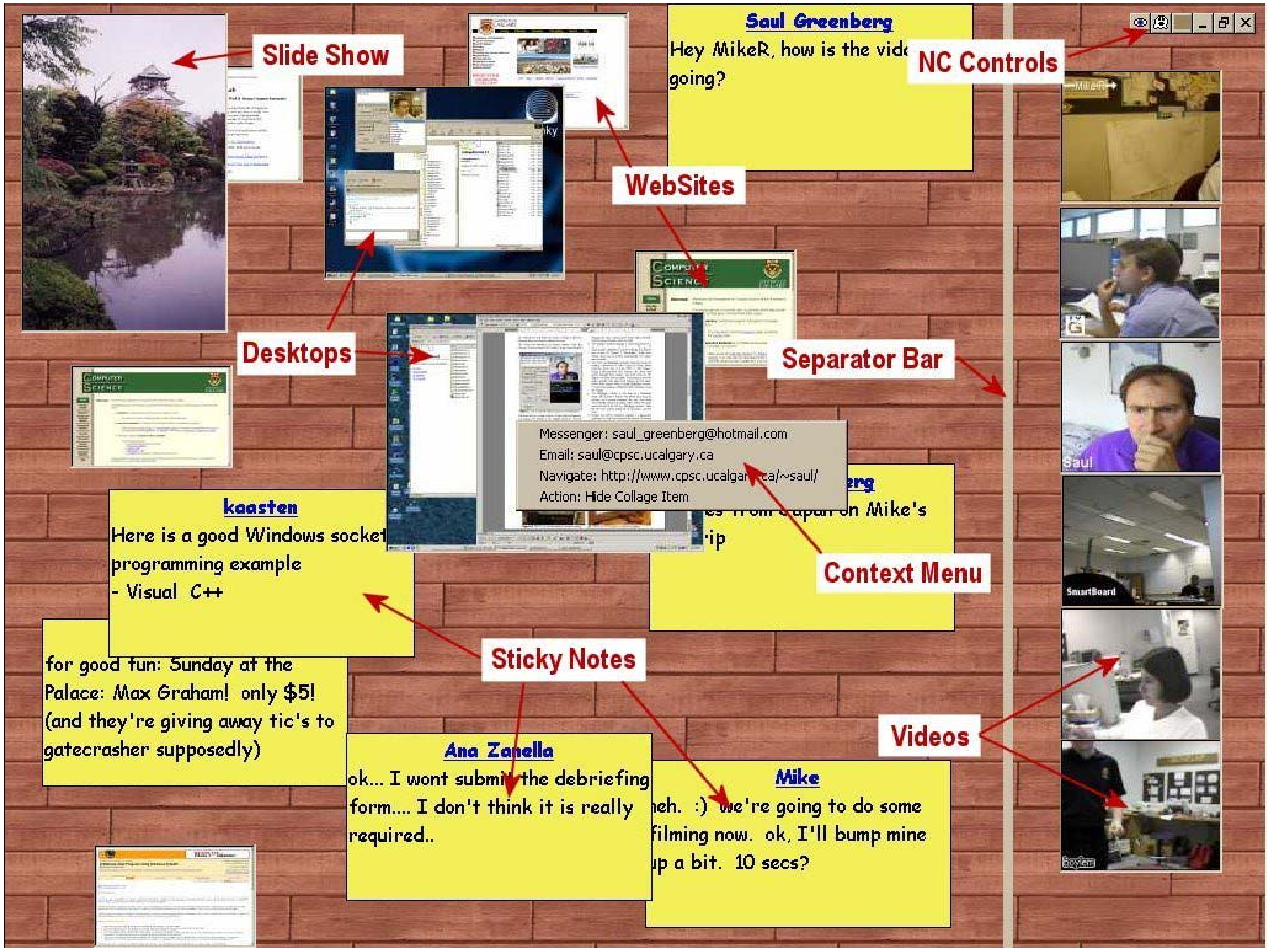


Figure 3.1 – A paper bulletin board illustrates a real world collage

This collage metaphor was chosen for several reasons. First, the overlap of items inherent on a large collage acknowledges that there may be a large number of information fragments, too many to fit neatly in the display space available (e.g., Figure 3.1). Second, collages are customarily used to present unstructured information comprising diverse media, conceding that awareness information comes in many forms (see Figure 3.1, again). Third, a collage gives a temporal history, with new items placed atop old ones. While newer items dominate, one can still see parts of the older ones. In the end, the metaphor allows a large amount of this awareness information, in the form of discrete and potentially diverse Media Items, to be displayed in the same space.


Slide Show
WebSites
NC Controls
Desktops
Separator Bar
Context Menu
Sticky Notes
Videos
Messenger: saul_greenberg@hotmail.com
Email: saul@cpsc.ucalgary.ca
Navigate: http://www.cpsc.ucalgary.ca/~saul/
Action: Hide Collage Item
MikeR
Saul
SmartBoard

Saul Greenberg
Hey MikeR, how is the vid
going?

on Mike's
ip

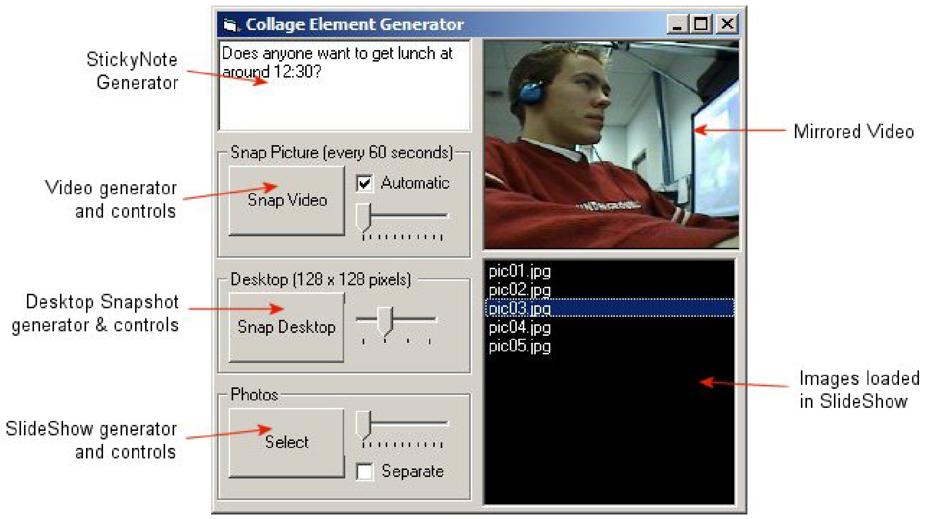
kaasten
Here is a good Windows socket
programming example
- Visual C++

for good fun: Sunday at the
Palace: Max Graham! only $5!
(and they're giving away tic's to
gatecrasher supposedly)

Ana Zanella
ok... I wont submit the debriefing
form.... I don't think it is really
required..

Mike
heh. :) we're going to do some
filming now. ok, I'll bump mine
up a bit. 10 secs?

Figure 3.2 – The Notification Collage


StickyNote Generator
Video generator and controls
Desktop Snapshot generator & controls
SlideShow generator and controls
Collage Element Generator
Snap Picture (every 60 seconds)
Snap Video
Automatic
Desktop (128 x 128 pixels)
Snap Desktop
Photos
Select
Separate
pic01.jpg
pic02.jpg
pic03.jpg
pic04.jpg
pic05.jpg
Mirrored Video
Images loaded in SlideShow

Does anyone want to get lunch at around 12:30?

Figure 3.3 - Media Item Client Interface

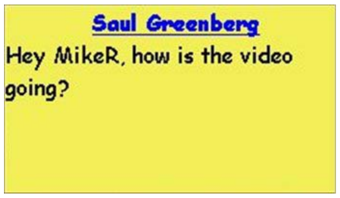
The initial vision was that the NC itself would live on a large, public display and that people would independently post Media Items to it from their personal workstations. This would leave the NC as a publicly accessible multimedia bulletin board, the idea of which was to promote informal awareness and casual interaction in our community of intimate collaborators. An initial prototype was built by Saul Greenberg, which was taken over and extended by me.

The rest of this section looks at particular features of the NC: its architecture, its media items, its display, and its user interaction.

3.1.1 Media Items

This initial NC deployment comprised separate clients that people would use to generate media items (e.g., Figure 3.3), and the NC client that displayed the items (Figure 3.2). Initially, six main media items were developed, all illustrated and labelled appropriately in Figure 3.2, and individually below:

a. The *Sticky Note* item followed the metaphor of a real world sticky note (e.g., a Postit™ note). When someone created a note via their client, it appeared on the NC with their name as a caption. As its owner typed their message (Figure 3.3, top left), text appeared within the NC note for others to see: edits are shown in real time. Note that only the owner could edit this message.



Saul Greenberg
Hey MikeR, how is the video going?

Sticky Note

b. The *Video* item showed a live video snapshot captured from the client via a desktop camera. Its owner used a slider to specify a refresh rate on the image ranging from 'near-live' to 'once per minute' (Figure 3.3, 2nd row & video mirrored on the upper right). By pressing the 'Snap Video' button, the owner could deliberately take and broadcast a video snapshot at any time.

Video Snapshot

c. The *Desktop* item displayed a thumbnail image of its owner's digital desktop. With the client (Figure 3.3, middle), the person can explicitly capture and post one's desktop at a desired image size (specified by a slider) to the NC.

Desktop Snapshot

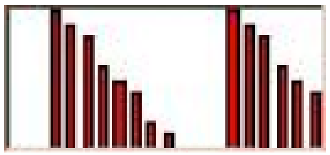
d. The *SlideShow* item extended the notion of tacking a picture to a bulletin board. Using the client (Figure 3.3, bottom left and right), a person could select one or more photos through a standard file dialog box and post them to the NC. Desired sizes were adjusted using a slider. Its owner could also specify whether the photos would cycle within a single *SlideShow* item on the NC, or as multiple *SlideShow* items (each containing a single photo) across the collage.

Slide Show

e. The *WebPage* portrayed a web page as a thumbnail image. A custom web browser (not shown) was employed to browse and post image links to the NC. Using this browser, its owner would navigate the web and as a side effect have webpage thumbnails created for each page visited. These were in turn posted to the NC as *WebPage* items. Anyone using the NC could use the context menu of a *Web Page* item to visit the page depicted by the thumbnail.

Web Page

f. Finally, the *Activity Indicator* displayed a continuously updating bar chart reflecting the amount of physical activity at its owner's workspace. A client (not shown) collected this information by using a physical proximity sensor to monitor movement in the space.


Activity Indicator

The NC was not limited to just these Media Items. With some programming effort, it was possible to add new ones, but it required source-level modification, recompilation, and redistribution of the NC display board if existing NC functionality did not adequately support the presentation of the new media item information.

3.1.2 Display

The NC was originally envisioned to run on a large public display in a common area (Figure 3.4a). This had certain implications in terms of how it would be designed. As it was seen residing on a large public display, items were designed to be static in size, roughly the same size as items on a real world bulletin board when projected on the large screen display. Font size on *Sticky Notes* was set so that they could be read on the public display at a modest distance, but not so large that it unreasonably limited the amount of messages that would fit. With these item sizes and the public display in mind, the NC itself

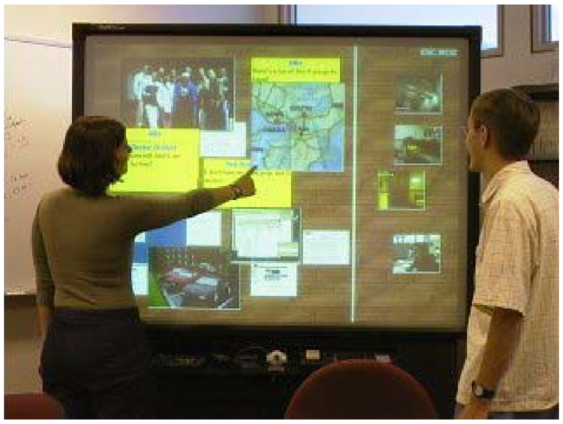


Figure 3.4a – Public Display

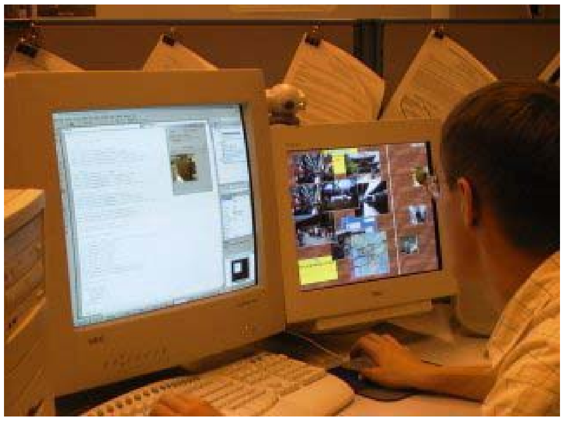
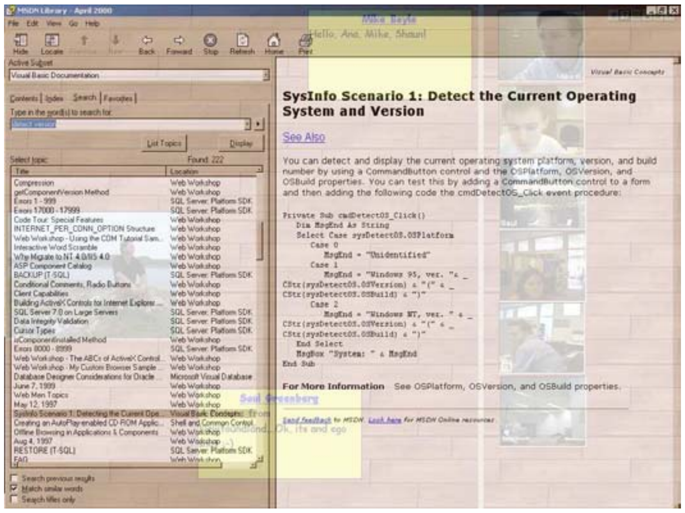


Figure 3.4b – Personal Display


MSDN Library - April 2000
File Edit View Go Help
Hide Locate Back Forward Stop Refresh Home Print
Active Subset
Visual Basic Documentation
Contents Index Search Favorites
Type in the word(s) to search for
List Topics
Display
Select topic:
Found: 222
Mike Boyle
Hello, Ana, Mike, Shaun!
Visual Basic Concepts
Saul Greenberg

Title	Location
Compression	Web Workshop
getComponentVersion Method	Web Workshop
Errors 1 - 999	SQL Server: Platform SDK
Errors 17000 - 17999	SQL Server: Platform SDK
Code Tour: Special Features	Web Workshop
INTERNET_PER_CONN_OPTION Structure	Web Workshop
Web Workshop - Using the COM Tutorial Sam...	Web Workshop
Interactive Word Scramble	Web Workshop
Why Migrate to NT 4.0/IIS 4.0	Web Workshop
ASP Component Catalog	Web Workshop
BACKUP (T-SQL)	SQL Server: Platform SDK
Conditional Comments, Radio Buttons	Web Workshop
Client Capabilities	Web Workshop
Building ActiveX Controls for Internet Explorer ...	Web Workshop
SQL Server 7.0 on Large Servers	SQL Server: Platform SDK
Data Integrity Validation	SQL Server: Platform SDK
Cursor Types	SQL Server: Platform SDK
isComponentInstalled Method	Web Workshop
Errors 8000 - 8999	SQL Server: Platform SDK
Web Workshop - The ABCs of ActiveX Control...	Web Workshop
Web Workshop - My Custom Browser Sample ...	Web Workshop
Database Designer Considerations for Oracle ...	Microsoft Visual Database ...
June 7, 1999	Web Workshop
Web Men Topics	Web Workshop
May 12, 1997	Web Workshop
SysInfo Scenario 1: Detecting the Current Ope...	Visual Basic Concepts
Creating an AutoPlay-enabled CD-ROM Applic...	Shell and Common Control
Offline Browsing in Applications & Components	Web Workshop
Aug 4, 1997	Web Workshop
RESTORE (T-SQL)	SQL Server: Platform SDK
FAQ	Web Workshop

☐ Search previous results
☑ Match similar words
☐ Search titles only

SysInfo Scenario 1: Detect the Current Operating System and Version

See Also

You can detect and display the current operating system platform, version, and build number by using a CommandButton control and the OSPlatform, OSVersion, and OSBuild properties. You can test this by adding a CommandButton control to a form and then adding the following code the cmdDetectOS_Click event procedure:

```
Private Sub cmdDetectOS_Click()
   Dim MsgEnd As String
   Select Case sysDetectOS.OSPlatform
      Case 0
         MsgEnd = "Unidentified"
      Case 1
         MsgEnd = "Windows 95, ver. " & _
CStr(sysDetectOS.OSVersion) & "(" & _
CStr(sysDetectOS.OSBuild) & ")"
      Case 2
         MsgEnd = "Windows NT, ver. " & _
CStr(sysDetectOS.OSVersion) & "(" & _
CStr(sysDetectOS.OSBuild) & ")"
   End Select
   MsgBox "System: " & MsgEnd
End Sub
```

For More Information See OSPlatform, OSVersion, and OSBuild properties.

Send feedback to MSDN. Look here for MSDN Online resources.

Figure 3.4c – Ghosted Display

was designed as a full-screen application. It did not have a title bar and it hid the standard Windows task-bar.

As described in our initial experiences below, the NC quickly found its way onto peoples' personal workstations. If they had a single display, the NC would compete with the user's normal productivity applications for space. Since the NC is a 'secondary' information source, this often meant it was buried by other application windows. In the ideal case, people displayed the NC on a second monitor (Figure 3.4b). This second display was handy because of the rather large desktop footprint used by NC. To attempt to ameliorate the amount of space it required on a person's desktop, a 'ghosting' feature was added, where a user could click a button and the NC became translucent (Figure 3.4c). As a semi-visible, click-through layer on top of other windows, the NC could be passively tracked without loss of application work space. However, the main problem remained that the NC did not scale well (overcrowding of media items was more severe when the NC window was smaller) and even on a second monitor, the full size NC still represented a significantly large scene that competed for the user's attention.

3.1.3 Aging and Competition

When new Media Items first appeared on the NC canvas, they were randomly positioned on the left side of the vertical separator bar, always on top of other items that happen to occupy the same space. As the number of posted items increased, new items eventually covered old ones. When no new items had appeared for a certain amount of time, the NC would occasionally 'bubble up' old items towards the surface. The result was that new items were almost always visible for a reasonable amount of time, giving them a good chance of being noticed. The appearance, eventual decay, and occasional reappearance of items reflected that some (but not all) information is timely yet ephemeral, that some information may be missed, and that peoples' awareness of information fragments is sometimes accidental and serendipitous. This again followed the analogue of the real life bulletin board where not every notice posted will be attended to by each person viewing the board.

3.1.4 Adjusting Item Visibility and Salience

When a person attended to the NC, they could increase the visibility and salience of particular media items. They could select individual Media Items to raise them back to the surface, drag items around on the canvas to reposition them, or hide items entirely from view via a menu option. Items could also be dragged over to the right side of the vertical separator bar (Figure 3.2). Because the NC positioned new items only on the area to the left of the vertical separator bar, placing an item on the right meant that it would never be covered by the automatic positioning of NC items. People could slide this separator bar right or left, adjusting the proportion of the canvas dedicated to newly arriving items and items they deemed more important. When this was done, items already on the canvas were automatically shifted proportionately to fill or compress into the re-allocated space.

All of these actions affected only the local NC client, and not the view of other clients connected. Thus the NC was a relaxed 'what you see is what I see' system (Stefik, Bobrow, Foster, Lanning & Tatar, 1986); while the same information was available to all connected NC clients, individuals could adjust each item's salience to their own taste. This allowed different people to track different sets of information that they considered more individually, and this increased the likelihood they would notice changes to items more closely affecting them.

3.1.5 Acting on Information

A person could act on any Media Item by right-clicking it with their mouse and raising a context menu (Figure 3.2). Through this menu, people could respond to whoever posted the item by emailing or instant messaging them, or they could visit the homepage of the item poster. In addition to these standard contact methods, some items also provided their own custom interactions. For example, a person could select an additional item in the *WebPage* item's menu which would raise a web browser and automatically navigate to the actual site depicted by the thumbnail.

3.1.6 NC Architecture

The Notification Collage was built atop a client / server architecture as illustrated in Figure 3.5. At its heart was a Shared Dictionary server that maintained a cache of string-based key/value pairs, a client-side library (DLL) that handled all communication with the server, and a simple API for programming clients. A typical sequence illustrates how this architecture worked (numbers match those in Figure 3.5 and using as an example an image captured by a client and distributed to all others).

1. Through the API, *input clients* (posting client programs) announce (or publish) new or altered key/value pairs, or requests to remove a key and its associated value.
2. The client-side DLL marshals this announcement as a string containing the command, the key and the value.
3. The DLL sends the marshalled string to the server via a socket connection.

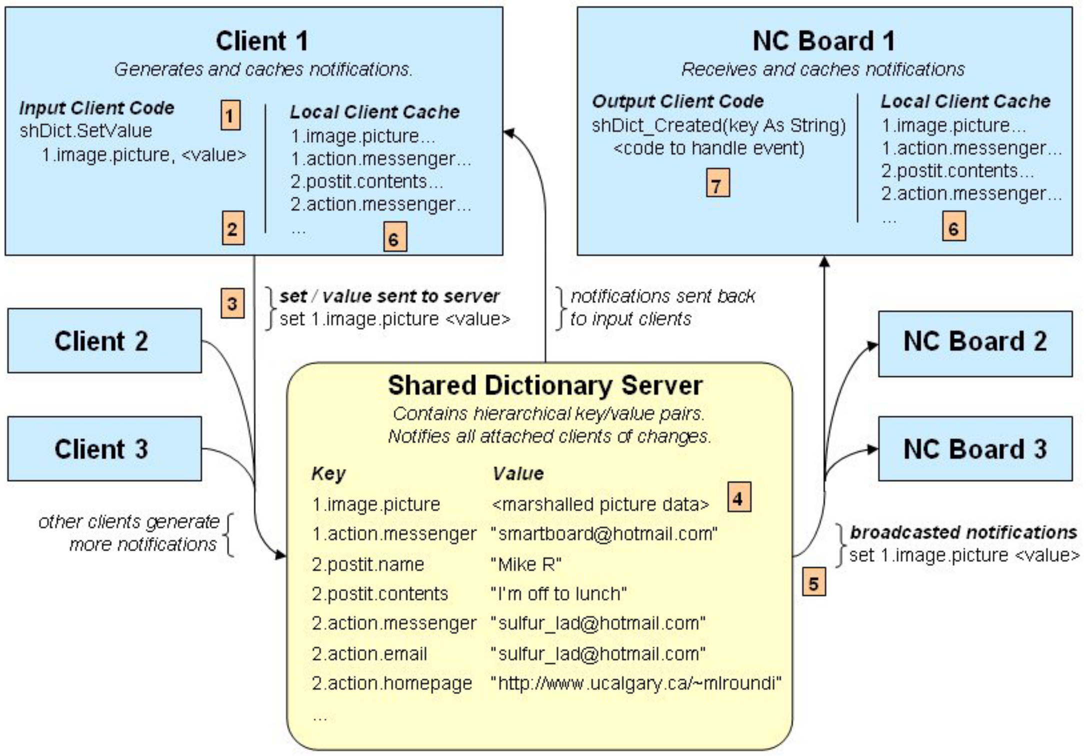


Figure 3.5 – Notification Collage Architecture

4. The server updates its local shared dictionary cache as directed.
5. The server then broadcasts the same sequence to all connected clients (including the posting clients).
6. Upon receipt of this message, each client-side DLL updates its own local cache of the shared dictionary.
7. The client-side DLL generates an event in the *output clients* (NC Boards) that signals via the API that a particular key has been created, modified, or destroyed.

This architecture was similar to a notification server (Patterson et al, 1996), which also rebroadcasts incoming messages. There were several key differences. The maintenance of all key/value pairs within the shared dictionary meant that new clients arriving after some activity had already occurred could request the current status of the dictionary (which happened automatically upon connection). This in turn meant that the client could update its display as needed. That is, it served as a distributed model-view controller (Stefik et al, 1986). In contrast, clients of a pure notification server must use some other mechanism to get the current state e.g., by requesting it from another client (Greenberg & Roseman, 1999). A cache was used so that client requests for a key's value did not require another trip to the server, thus speeding updates.

Keys were hierarchical, which meant that clients could structure information they were sending out, and could selectively pattern-match incoming events (Greenberg & Roseman, 1999). NC exploited this by setting the following convention for naming keys: each media item was identified by a unique id (a number) followed by a set of words that indicated what information the value would contain. For example, a *Sticky Note* item (see Figure 3.2) used five keys:

- <id #>.sticky.name indicated the name of the note's creator, as posted as a caption on the top of the sticky note
- <id #>.sticky.contents held the note's contents, which change as the owner types their message in their client

- <id #>.action.messenger held the creator's instant messenger contact name
- <id #>.action.email held the creator's email address
- <id #>.action.homepage held the creator's homepage URL

The first two keys were specific to the *Sticky Note* item. Any change to a note just required a change of the value of key <id #>.sticky.contents; the rest of the information (for example <id #>.sticky.name) did not have to be retransmitted. The three remaining keys used the sub-key name 'action' (e.g., <id #>*.action.*messenger) to indicate that they were part of the context menu for that media item (Figure 3.2).

Other media items worked in a similar fashion. All image-based items, for example, used the key <id #>.image.picture to hold a marshalled version of the image. Finally, reusing media items was easy because individual items were associated with a key's id. Changing <id #>.sticky.contents told the receiver to change the contents of one particular *Sticky Note* item instance (e.g., as would happen when someone typed text into a *Sticky Note*). Similarly, continuously changing <id #>.image.picture would create an image stream within a single item (this is how the *Slide Show*, the *Desktop*, and the *Video* items were implemented). When keys were received that contained a new id number, the NC client created a new media item that matched the key's type.

In summary, the NC worked by having all clients publish key/value pairs describing a media item (embedded in key names), its information (through values stored at key locations), and how someone could respond to it (with keys containing the sub-key 'action'). When NC received these as events, it read the keys and associated information, and constructed or modified the media item on the board to reflect this information. Since a key name convention was agreed upon, creating new media items was relatively straightforward. A new client for generating the media item information was built, and code was added to the NC client to display the new item.

3.2 Initial Usage Experiences

The first version of the NC was created by Saul Greenberg and evolved by myself in May of 2000 (Greenberg & Rounding, 2001). Because it was new software still under active development, it was initially deployed only to our own research group. Experiences in this environment are summarized here.

3.2.1 The group and its setting

Our group comprised mostly research assistants, graduates and faculty: only two of us were initially involved in the NC project. Much of the group inhabited a research laboratory (floor-plan illustrated in Figure 3.6). One side of the laboratory comprised workstations and workbenches for graduates and research assistants. Computers were typically equipped with a camera and two monitors (Figure 3.4b). The other side of the lab contained a small, public meeting area containing a 72” rear-projected Smart Board, also equipped with a camera (Figure 3.4a). Workstation partitions in the lab were short, so people anywhere in the lab had some informal awareness of activity elsewhere in the room, including what was on the Smart Board (which directly faced the laboratory’s door). The Smart Board (which had its own ‘user identity’) always ran NC and a client that would post the video from its camera, showing a panorama of the room.

While some group members regularly worked in the laboratory, others did not. Faculty had separate offices. Other members were telecommuters. One faculty member regularly worked at his home office 110 km away, coming into the laboratory only occasionally. Some group members lived elsewhere; for example, one was on a work internship 1000 km away for part of the evaluation period. A few others alternated between work and home offices, with home use at its peak each day in the early morning and late evening hours.

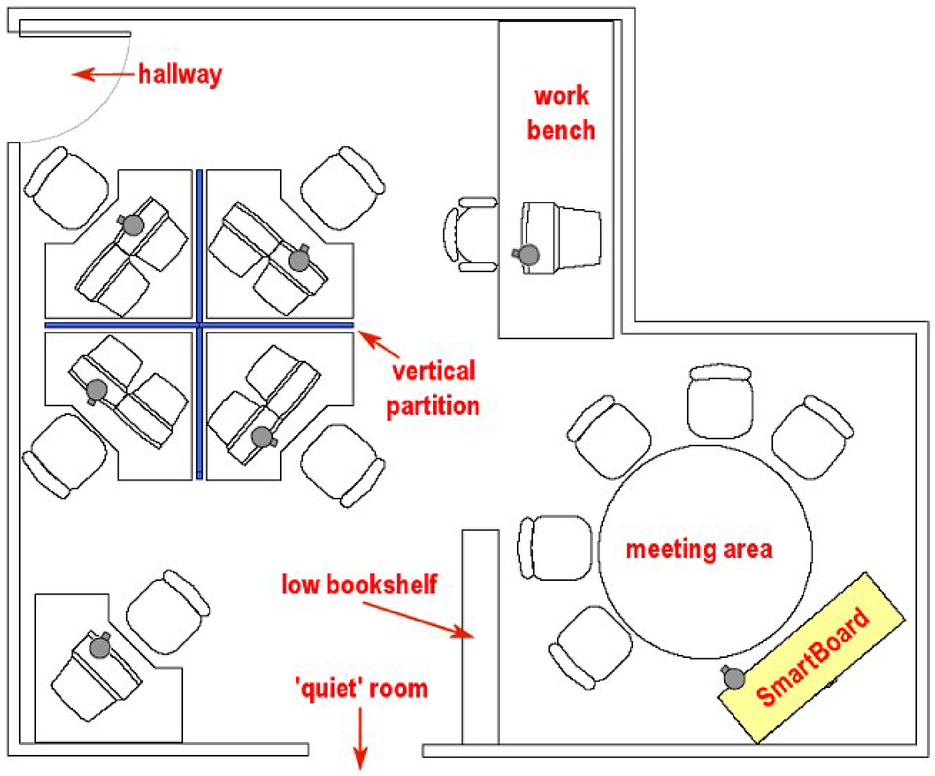


Figure 3.6 – Laboratory layout

3.2.2 Bootstrapping

The preliminary version of the NC was accessible only to co-located group members on a local network. Not surprisingly, there was very little activity on it since its users were in the same room and they could see each other directly.

When an internet-enabled version of NC was released (i.e., a version that allowed off-site people to connect to it), a dramatic change in its use was observed. There was a noticeable buzz of excitement: people wanted to join in, and those connected became evangelists for getting other team members to join. While the majority of users were still co-located, the ability to see and interact with remote telecommuters added significantly to their motivation and experience.

Peoples' first instinct was to post their visible presence to the NC via the *Video* item. This mimicked what is usually seen in most media spaces and instant messengers: all could project their presence to others and sense who was around. Video was considered special:

typically, people would move video images to the right side of their NC's separator so they would not be covered up. As telecommuters became visible and reachable through the NC, other people inside the laboratory became increasingly interested in using the NC; they would connect to it and stay connected to it.

3.2.3 Display

The initial expectation of NC was that people would run it only on the single large public display in the research laboratory, and that those outside the laboratory would connect to it every now and then from their personal machines. In practise, the NC quickly found its way to almost permanent display on everyone's desktop, even if those people were in the same room as the public display. We saw that people with multiple monitors did their main work on one monitor and had their view of the NC running on the other display: this accords with Grudin's findings that second displays are often used to hold peripheral information (Grudin, 2001). While those in the laboratory could glance around to the Smart Board, it was not always in their direct line of sight. They felt that having an instance of NC on their own machine made them more aware of changes, and they were better able to respond to particular events.

When the NC appeared on the public display however, people used it somewhat differently than when it appeared on their personal computers.

First, people not at their workstations used the public display as another means of tracking and posting information to the NC. This depended on several things: whether they were logged onto a workstation in the room, whether they were closer to the public display than to their workstation, and so on. As a result of the placement of the large display, some glanced at it briefly as they walked in the door. As could be expected, people would also bring other people's attention to items on the public display and would converse around them.

Second, telecommuters used the video generated by the camera attached to the Smart Board as a means to monitor and communicate with people seated at the meeting table or workbench, and with people wandering about the laboratory. This was a particularly important way for telecommuters to contact laboratory inhabitants when they were not at their workstations, as well as laboratory visitors and associates who did not have a personal NC. For example, we saw telecommuters notice and contact part-time members of the group who appeared occasionally in the laboratory.

Third, the Smart Board public display (a regular PC) became a more public artefact as a result of it being the NC's main residence. People began using it as a file server, placing files for sharing and broadcast, and it eventually did become the laboratory's official file server. This evolution of the NC server to a file server was due to people using NC as a medium to communicate amongst each other, and then using the file server in place of one of the missing features in NC: file sharing.

3.2.4 One-to-one, overheard and broadcast communication

People often took advantage of presence information by using the NC as an instant messenger. People were observed to initiate IM conversations by way of the context menu on a Media Item that invoked a one-on-one Microsoft Messenger session with the person who posted the item. More often however, people communicated within the NC itself through *Sticky Notes*. This direct contact generally began with the other person's name e.g., 'Hey Mike…' While not explicitly designed as a chat tool, people would frequently communicate to each other in real time by modifying their own note and by monitoring changes to others.

Conversations over *Sticky Notes* differ considerably from those over Microsoft Messenger. Since anyone connected to NC could see the *Sticky Note* contents, the conversation could be overheard. This became an important opportunity for casual interaction (Kraut et al., 1988) where others would serendipitously join the conversation. These others would sometimes just say hello, or would join in any random bantering, or would contribute to a purposeful conversation when they felt they had something to add.

Other people in the laboratory seeing incoming *Sticky Notes* also meant that they could tell others about messages directed to them. This happened when the addressee did not notice the note, or when he/she was a room visitor. For example, a note on the NC labelled “Hey Mike…” would be noticed by someone just arriving in the lab. They would then turn to Mike and say “Hey Mike, Saul’s trying to ask you a question…”

Sticky Notes were also used for purely broadcast communication. People used them to inform the group about current or upcoming events, to annotate other Media Items (e.g., annotating a *Slide Show* by posting a *Sticky Note* that says “Photos from my trip to Japan”), or to elicit group comments. *Sticky Notes* also served as a way to make general queries or requests which could be answered by anyone e.g., ‘Does anyone know…’

3.2.5 Video as conversation and opportunity

People used *Sticky Notes* and *Video* elements in tandem. They would sometimes wave to re-enforce a greeting, and would accentuate a note’s message by exaggerating their body language (laughter, thumbs up, looks of shock, making faces at one-another). They also used the video to show others physical things being talked about. One person, for example, used the video to display the covers of a large number of boxes of software they had just purchased.

Aside from knowing that a person was around, the group used other things they saw on the video as opportunities for conversation. For example, one conversation stemmed from seeing a particularly ugly hat a person was wearing, with quite a few people eventually joining into the teasing. Seeing ‘visitors’ in the video also led to many conversations. One telecommuter introduced his children (who were visible on the video) to other group members. In another case, a person working at home recognized a visitor to another person’s office that he had not seen for years. They began to chat in *Sticky Notes*. Since the telecommuter’s wife had also met this person several years back, she too came up and joined the conversation.

Video also provided situational feedback. In one case, a telecommuter asked a person in the laboratory to get something from elsewhere in the building. He saw her get up and leave, thus affirming that she was doing what was requested.

3.2.6 Artefact display

People used the NC for displaying other artefacts to the group as different Media Items. We saw one popular example where people would post digital photos to the NC via the *SlideShow* element for others to see. These included photos of personal vacations, families and friends, and group outings. Other examples included postings of large desktop thumbnails and individual photos. People sometimes included *Sticky Notes* to explain the slide show. Most importantly (and unlike instant messaging and media space system use) conversations would sometimes arise as a consequence of people seeing these artefacts.

3.2.7 Privacy Issues

Several privacy issues accompanied NC use. First, NC does not guarantee reciprocity. One can use it without signalling their presence through video. Alternatively, a person can have the video capture client running without having the NC displayed. While people did usually enforce reciprocity through social habit, we saw inadvertent reciprocity breakdowns. One example of this breakdown stemmed from a power-save facility that turned off the Smart Board projector after a certain amount of time. Telecommuters sending a message to a person captured by the Smart Board camera when the projector was switched off could then not make contact. Of course, that person did not see the message as the display was dark.

Occasionally, people captured by the *Video* element had no idea that their image was being broadcast. For example, one telecommuter reported seeing the lights come on after hours in the laboratory, and watching a cleaning person (unaware that she was being monitored) going about her duties.

Telecommuters who used NC video from home reported other privacy concerns. One telecommuter's home office doubled as a guest bedroom. While he felt video was essential (and had it always on), his wife (who was not part of this NC community) did not like the idea: she received no benefit from having the video on and saw it as a possible intrusion. The telecommuter was also more aware of his appearance: while he previously worked with his shirt off on warm days, he no longer did so. He was also concerned about inadvertently broadcasting situations visible within the room e.g., family members wandering by in various states of undress, or who used the room for other purposes. As a partial solution, he habitually rotated the camera to face out the window when leaving the room. Another telecommuter commented and apologized for the 'mess' visible in his room.

3.2.8 Distraction issues

When many items were on the NC, people found it more difficult to find information they felt important. While people could post many elements to the collage—slide shows, photos, videos—most felt video to be the most essential element. Although *Video* elements always rose to the surface of the collage on every update, it was still effortful to find them in a busy collage. Related to this, people sometimes wanted a way to 'filter' items from the display, especially if the NC was on their personal computer. For example, one person commented that he wanted to remove a *SlideShow* element from the NC because he found the current set of photos uninteresting and the cycling of images distracting. As a consequence, we added the option for people to 'hide' elements. We also saw people regularly move elements they felt important — particularly videos — to the right side of the NC's separator bar so they would not be covered up.

3.2.9 Summary of experiences

After reviewing these experiences, I saw that people treated the NC more as a virtual public room (Roseman & Greenberg, 1996) rather than a bulletin board encouraging interaction. One person would post a Media Item, and others would (eventually) become aware of it and selectively react to it. What typically ensued was a sometimes brief, sometimes lengthy, sometimes parallel interaction between many people at once on the board. People made faces at each other, chatted through the *Sticky Notes*, and often posted other items onto the NC that were relevant to the conversation. These experiences suggest that NC affords uses spanning several types of collaborative tools: awareness notifiers, instant messengers, media spaces, asynchronous bulletin boards, and MUDs. They also suggest different communication styles: broadcasts, playful, purposeful, casual, one to one, many to many, directed, and so on.

3.3 Discussion and revision

While the NC remains a successful system even in the state as described thus far, we found that actual use of the system did not quite match with our original vision of its use. In particular, we noticed usage patterns that indicated both strengths and weaknesses of the current system. In turn, this knowledge suggested ways to redesign the NC to accentuate the strengths and to more closely fit it to the actual way in which people were using it.

3.3.1 NC Strengths and Weaknesses

Our initial usage period showed that the NC was an overall success, promoting informal awareness and casual interaction within our group. However, evolution in the way people were using it led to numerous ways in which the system could also be improved. This section outlines both successes and weaknesses of the system, and makes several recommendations as to how the NC should be redesigned. In hindsight, some of these may seem obvious. However, it would have been extremely helpful to know these ahead of time.

1. The NC became useful when people outside the laboratory (or the immediate physical domain) could connect, enabling the co-located group to stay in touch with those not immediately present. *The NC should make connecting over the internet for both co-located and distance separated people easy.*

2. Running NC on the large public display was advantageous to visitors, to new arrivals, and to occupants of the laboratory. Visitors could get a good idea for what was happening in the group, new arrivals could gain an immediate partial understanding of the current group state, and normal occupants could apprise others of items of import

they hadn't yet noticed. *The NC needs to remain amenable to use on a large public display.*

3. When people started running NC on their personal workstations, they found that there was no way for them to accurately tell who else was running it and watching. However, people wanted to see others, and wanted others to be able to see them. Most posted video almost immediately, explicitly indicating presence. The few who did not have desktop cameras or who chose to not post their video could essentially connect and 'hide' (sometimes unintentionally). In turn, this could provoke feelings of being 'spied on' in those who did post video. *The NC needs to enforce at least a rudimentary sense of presence when people connect, enabling everyone to see 'who is around'.*

4. Using media items enabled people to post personal artefacts for others to see and share, and gave opportunities for others to post their own in turn. Further, running NC on their personal displays allowed people more immediate access to the artefacts posted by others. *Media items need to be supported and used because people use them effectively to post and share information artefacts and act on them through the NC.*

5. People organized media items to their liking, as it enabled them to increase the visibility of some pieces of information over others. Media item movement and hiding was available. While movement was easy by simply clicking and dragging an item, hiding an item required a user to find its context menu and select the appropriate function there (this action proved difficult on the early versions of our touch-sensitive SmartBoard). *The NC definitely needs to make easy support for managing media items available on both a per session and per media item basis.*

6. People consistently used *Sticky Notes* on the NC to act out conversations. Making these conversations visible to the group by posting them to NC meant that they could be 'overheard'. Thus anyone could monitor and join in on interaction, and people could inform their peers of notes directed specifically to them. Also, unlike instant messaging and media spaces, conversations sometimes began from people seeing interesting artefacts within the space and wanting to talk about them. Examples include an interesting *Web Page* or set of photos in a *Slide Show*. *The NC should support this*

information broadcasting style as a communication metaphor because it fits well in a community of intimate collaborators.

7. Media Items were used in tandem to augment each other. For example, a face in a *Video* was used to accentuate a textual statement on a *Sticky Note*, or a *Sticky Note* could be used to describe a series of pictures in a *Slide Show. The NC should make it possible to quickly identify particular individual media items, making it easier to draw attention to specific items.*

8. While people wanted to run the NC on their personal desktops, they could not manipulate anything on it besides media item position and visibility. The separate client-side interfaces along with the NC itself meant it was a large-footprint display, and that there was high user overhead to start the NC and its various clients. There was definite concern about the trade-off between screen space and the value of the information on NC. However, because people wanted to be able to 'see each other' properly, running it locally on a workstation was seen as necessary. In order for a conversation to occur, people had to switch between at least two interfaces (for example, a *Sticky Note* client to broadcast their message and a NC client to review others' reactions). This could get even more complicated when using several items to engage in conversation. For example, using a *Sticky Note* to describe a series of photos in a *Slide Show* with an accompanying *Web Page* containing a trip diary and a *Video* to display a very happy face could have involved up to five separate applications, depending on which posting clients were used. This overhead was excessive. *The NC board itself should be used for input and output, supporting direct in-place posting and editing of media items and their contents.*

9. For development purposes, using the simple shared dictionary server and an agreed upon key structure to represent the contents of the NC was successful. Media items could be differentiated in the server from one instance to another, and any amount of information could be stored online for each. Event notifications from specific items were easy to retrieve and handle, and new NC arrivals could always retrieve the current contents of the NC to become up to date. In this way, NC effectively helped people who were

separated from the main group stay in contact. However, conversations were frequently interrupted by the shared dictionary server crashing. This left all clients hanging, and people were forced to restart, and re-connect their NC software (both item posting clients and the NC client itself). *A robust Internet communication layer, modelled after the shared dictionary architecture used in the initial prototype should be developed for the NC.*

10. Asynchronous communication using NC was possible by leaving a Media Item posting client running. This meant however, that for an item to live beyond working hours, the posting user needed to leave their NC item client on, plus their workstation on and logged in. While a 'hack' was written so some versions of Media Item clients would leave items on NC after they were closed, it was quickly discovered that there was then no way to ever remove them from the NC at all, producing excessive clutter. *The NC should support offline asynchronous communication by allowing people to leave items up for a duration that is independent of login status with the shared dictionary server removing information needed for expired items.*

11. From the development point of view, extending the NC with new media items was possible, but was a cumbersome experience. Extension required source code access to the NC for the purpose of adding the new item. If several people were developing new media item types independently, then several different code versions were circulating and needed synchronizing each time a new item was finished, even before it was released for people to try. *The NC needs to have an architecture to separate media item development from media item display and use (or alternatively, from NC development).*

12. As mentioned in the previous point, the NC was successfully extended with new media items. However, it was a cumbersome experience for users to include them. They had to remove NC from their system, acquire a new version that supported the new item, and acquire the item's posting client. Instead of extending NC in this fashion, in the end several items were developed to simply emulate others to avoid the process (for example, *Slide Show* and *Videos* all actually use a *Slide Show* item). *The NC needs a* *system for extending media items where people can take advantage of the new items immediately with minimal effort.*

3.3.2 Summary

This review of our experiences and the resulting strengths and weaknesses leads to a new design rationale for the NC. The single design intent was that NC would be a publicly displayed multimedia bulletin board that could be posted to by people running simple client programs. Table 1 (below) summarizes a set of important revision recommendations for the NC based on initial usage experience.

Based on these experiences and observations, I proposed a new design rationale for a modified (albeit completely re-implemented) Notification Collage. This design rationale is split into two parts, the first concentrating on the user experience and the second on the media item developer experience.

1. *NC user experience design:* The NC is a publicly accessed multimedia bulletin board. It will run on both a large public display visible in a public space, and also on personal displays such that individual users can run it on their workstations (preferably on a second monitor). It will incorporate a single client interface where people will be able to post and interact with media items in-line.

2. *NC development and extension:* There can and will be more than one developer extending the NC with new Media Items. Integrating extensions needs to be seamless to those using the NC, so that they can continue to use NC while quickly gaining the benefit of new items. Media items will be uncoupled from the NC interface completely by designing an architecture to fit in between the two.

Table 1 – Notification Collage Redesign Recommendations

The NC should make connecting over the internet for both co-located and distance separated people easy.
The NC should be amenable to use on a large public display.
The NC needs to be able to enforce at least a rudimentary sense of presence when people connect, enabling everyone to see 'who is around'.
The use of discrete media items needs to be supported.
The NC needs easy support for managing media items available on both a per session and per media item basis.
The NC will follow a broadcast style communication metaphor.
The NC should make it possible to quickly identify particular individual media items.
The NC should be both an input and output interface. This will allow in-place posting and editing of media items and their contents.
The NC should exploit a robust client / server system for the shared dictionary architecture to support real-time interaction over the internet.
The NC should allow people to leave items up for a duration that is independent of login status.
The NC needs to have an architecture that separates media item development from NC development.
The NC needs a system for extending media items where people can take advantage of the new items immediately with minimal overhead.

The following chapters focus on each of these visions in parallel. Chapter 4 presents user experience of the redesigned NC, including its stock set of Media Items and the NC interface itself. The ability to include new media items onto the NC is also presented from a user's point of view. Chapter 5 discusses the technical aspects of the redesigned NC development and extension. This presents the architecture for the relationship between Media Items and the NC, as well as the deployment strategy from a technical viewpoint.

It is important to note that although presented sequentially, these chapters should be taken in parallel. The evolving visions were co-developed, where the concepts in one interplayed with the other.

Chapter 4. The New Notification Collage User Experience

In chapter 3, I introduced the Notification Collage (NC) and our initial development and usage experiences with it. Several recommendations for the redesign of the Notification Collage came out of those initial experiences. The most important redesign idea was to merge all the various clients – the Notification Collage board, the various media item clients that individuals use to capture and present information – into a single, unified user interface. In this chapter, I present the NC board interface as redesigned and implemented concurrently with the *Media Item Architecture* (presented in chapter 5) as a single unified user interface. I first revisit the redesign recommendations of chapter 3 to define three goals for the new user interface. I then break down these goals by discussing the re-implemented NC user experience, including a discussion of the stock set of media items included with the NC board. While interface features hint at the new technical aspects of the NC, I defer a detailed description of its architecture until Chapter 5.

4.1 Redesign Goals

The design recommendations of chapter 3 relating to the user experience have been consolidated into three goals that will be addressed in this chapter:

1. **The NC board should allow people to rapidly connect and use a public space that incorporates both basic presence information of other connected people and a collage of media items.** The collage metaphor of randomly placed media items will be maintained to keep the NC amenable to large displays as well as personal displays. In addition, the new interface will include a generic presence indicator, allowing those

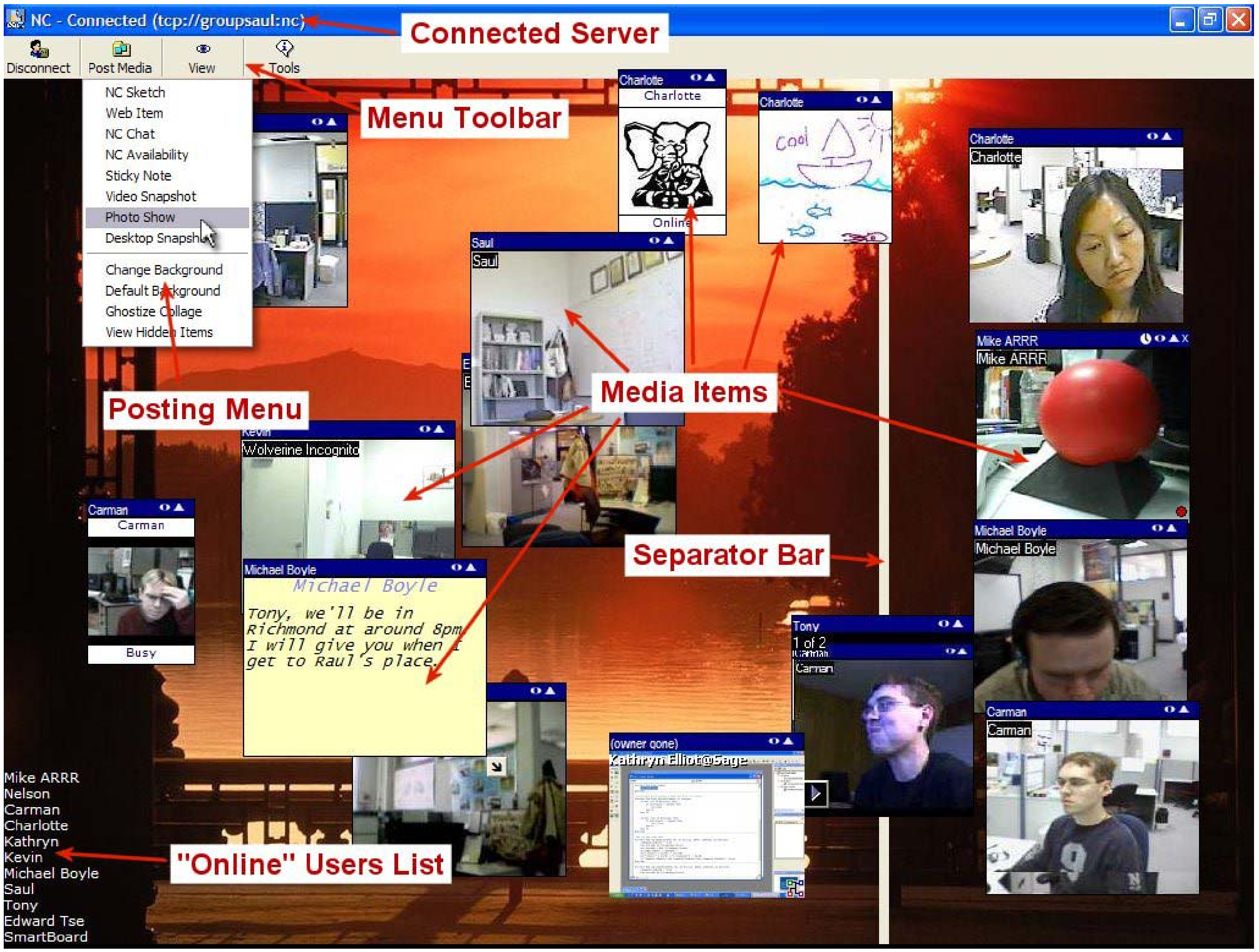
connected to see ‘who is around’ (via their login status) and to discover more information about those people.

2. **People should be able to add and manipulate media items directly using a single user interface.** The interface components that existed as separate clients for posting and viewing media items will be consolidated into one single interface encompassing all interaction using the *Media Item Architecture* (presented in chapter 5). That is, media items on the NC will not only display information but will also provide their own user interfaces for information capture.

3. **The NC board should include a stock set of media items designed to support informal awareness leading to casual interaction and / or information sharing in a group of intimate collaborators, as well as the means to incorporate items.** Media items should include the successful ones as presented in chapter 3, but enhanced to provide richer interaction, as well as several new items that are expected to be useful.. All media items should ideally give people not only awareness, but should lead to information sharing and / or casual interaction. Of course, we cannot anticipate what items – especially ones that serve a group’s special purposes – would be useful. Consequently, the NC Board should have a means for a person to include new items developed by others, and for others to add that new item to their own NC board.

4.2 The NC Board Interface

The NC board client is the application window that contains media items. While there are many ways the NC board could display these media items, our choice was to keep the collage metaphor for the new version (Figure 4.1), as we deemed this successful in the original prototype. The most notable cosmetic differences between the new and the initial NC boards are the addition of the standard Windows title bar and the menu toolbar at the top of the window, and the user list in the bottom left-hand corner (see Figure 4.1). In this section, I will describe the new NC board and how it mimics and differs from the initial prototype.



Michael Boyle

Tony, we'll be in Richmond at around 8pm I will give you when I get to Raul's place.

Figure 4.1 - The Notification Collage Redesigned

4.2.1 The NC Board Revisited

At a high level, the new NC board client (Figure 4.1) looks much the same as the initial version presented in chapter 3. The same canvas is displayed, along with the vertical separator bar. As with the original NC, newly appearing media items will always be placed randomly to the left side of the vertical bar. Placing items on the right side of the bar still means that those chosen items will never be covered by newly arriving ones, allowing people to set aside certain media items. Sliding the vertical bar left and right will grow and shrink the areas dedicated to newly arriving items and highlighted items. The ghosting feature also still exists, and appears in the ‘View’ menu along with the ‘show all hidden items’ tool, both selectable from the toolbar at the top of the NC board (Figure 4.1, top). Unlike the previous version, a person can now customize the board’s appearance by selecting an image to use as the NC board’s background ‘wallpaper’ from the ‘Tools’ menu. The remaining sub-sections will describe other key differences.

4.2.2 Connecting

With the original NC, connecting to the server was synonymous with users creating and destroying the NC application. While this worked, the scheme did not recognize that:

a) people may want to disconnect for a short time from the NC without stopping it (e.g., to reconnect later);

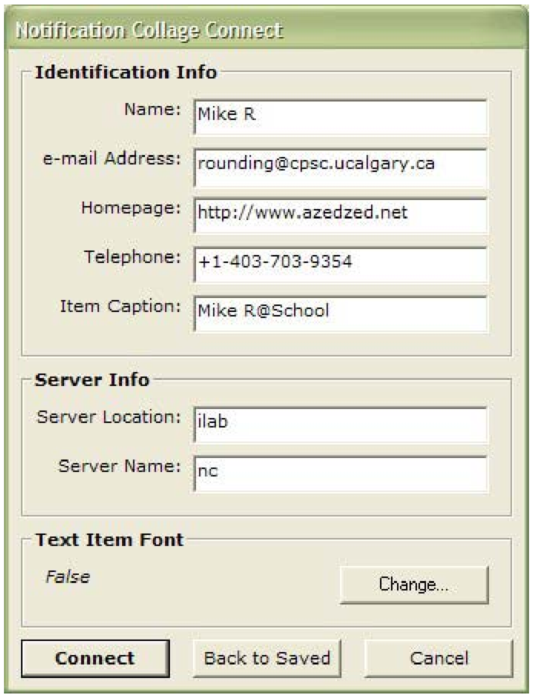
b) people may want to switch from the current community to another one (i.e., a disconnect followed by a reconnect to a different server);

c) the connection may be dropped by the system (e.g., due to a network bottleneck) and people would want to reconnect back as soon as possible.

To make it easy for people to manage their connection status, a ‘Connect/Disconnect’ button was added to the NC board on the new toolbar (Figure 4.1, top left).

When the ‘Connect/Disconnect’ button is pressed, the NC board displays the connection dialog (Figure 4.2) which collects some basic information from the person. First, the ‘Identification Info’ (Figure 4.2, top section) is used to specify the local person’s contact information to the server. Second, ‘Server Info’ (Figure 4.2, middle section) is used to specify the community’s server, including its location and its name. Third, the ‘text item font’ section (Figure 4.2, bottom section) allows the user to specify a font which is used as the default by text-based media items (for example, the *Sticky Note*).

The buttons at the bottom of Figure 4.2 control the outcome of the dialog box. First, ‘Connect’ attempts to connect to the server which, if successful, will display all media items currently posted. Second, as a side effect of connecting, all the user information that was entered or edited in the dialog box is also saved to the Windows registry file. This saved information is automatically loaded and redisplayed every time the person wishes to connect, so that retyping is not needed. If desired, this information can be changed at any time by editing it directly, although one can ‘undo’ by reverting to the last saved


Notification Collage Connect

Identification Info

Name: Mike R

e-mail Address: rounding@cpsc.ucalgary.ca

Homepage: http://www.azedzed.net

Telephone: +1-403-703-9354

Item Caption: Mike R@School

Server Info

Server Location: ilab

Server Name: nc

Text Item Font

False

Figure 4.2 – The NC connection dialog

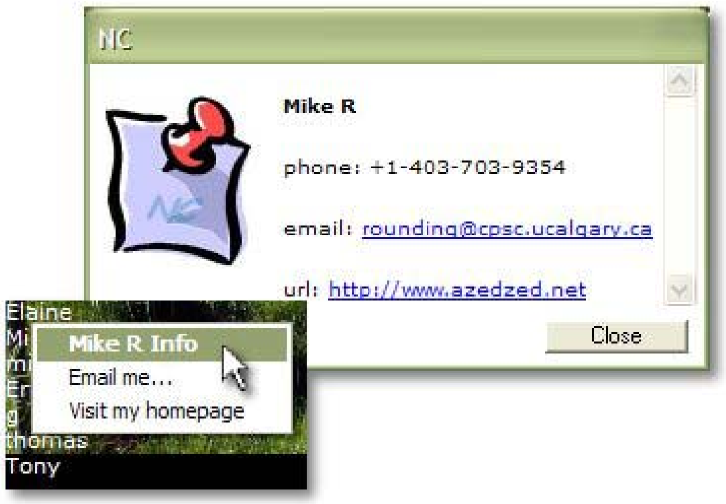
information through the ‘back to saved’ button (Figure 4.2, bottom). ‘Cancel’ closes the dialog box without connecting or saving the user information.

Upon successful connection, the NC board’s title bar displays the name and location of the server that has been connected to (Figure 4.1, top). A ‘Disconnect’ button appears in place of the ‘Connect’ button and allows a person to disconnect from the server, which closes all media items to show the empty NC board with the ‘Connect’ reappeared. This allows people to disconnect from one NC session and connect to another one on a different server without restarting the application. Existing media items are displayed, along with a list of all connected users (Figure 4.1, bottom left and Section 4.2.3).

Finally, the NC is further tuned to recognize dropped connections. When this happens, a dialog box (not shown) is raised, asking the user whether to reconnect or not.

4.2.3 Presence

One of the side-effects of the initial NC board was that connected people could ‘hide’ from others (intentionally or not) if they did not post any media items. Similarly, people were only reachable through the posted media items as their contact information was provided on a per-media item basis only.


NC

Mike R

phone: +1-403-703-9354

email: rounding@cpsc.ucalgary.ca

url: http://www.azedzed.net

Figure 4.3 – Accessing contact information

With the new NC board, the contact information provided by people in their connection dialog (Figure 4.2) is automatically broadcast and stored on the server, and then sent to all NC board instances to create an 'online users' list of people. This list is displayed directly on the NC board and independent of media items (Figure 4.1, bottom left). The list provides some sense of presence for everyone who is connected, regardless of what media items are posted. As Figure 4.3 illustrates, a person can click each user's name in the list to gain access to the contact information they posted through their connection dialog box to email them directly and to link to that person's homepage on the web.

4.2.4 Media Items

As the most important goal for redesigning the NC board is to create a single interface for posting and manipulating media items, several features have been added to help people manage media items directly. As we will see, people not only control item position and visibility as in the initial version, but they now create and delete their own individual media items, control media item lifetime, and install new media items on the fly directly through the NC board.

Creating media items is now accomplished by selecting an item type from a menu accessed either through the 'Post Media' button on the toolbar (Figure 4.1, top) or by right-clicking the NC board canvas (Figure 4.4). When a media item (no matter what type of item it is) appears on the NC board, it does so within a rectangle that resembles a small window. This enclosing rectangle, or *media item container,* includes a small caption and title bar (outlined in Figure 4.5). The title bar has two different modes, depending on who owns the media item. For the *item poster*, all of the media item management functions are available (Figure 4.5a). These include two different options for removing media items from all connected NC boards (2 and 5 in Figure 4.5a) and functions for hiding the media item entirely from view or to shrink it to just the title bar

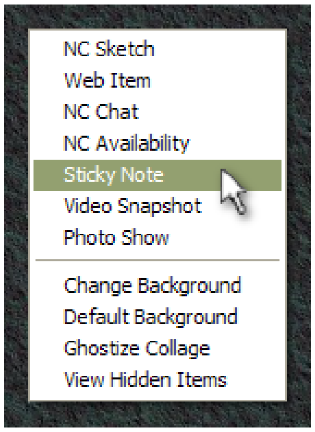
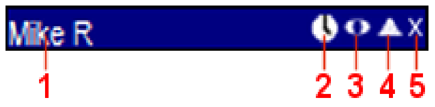
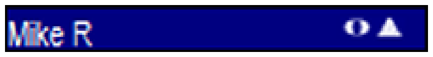


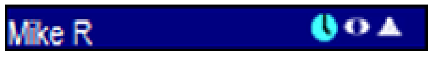
Figure 4.4 – Media Item Posting Menu



A: An owner's title bar and controls
1. name of item owner
2. timeout control icon
3. hide item
4. shrink to title bar, 'window-blind'
5. close item



B: Viewer's typical title bar controls



C: Viewer's title bar controls for an item showing timeout notification

Figure 4.5 - The media item container's title bar

(*aka* 'window-blinding') within the local NC board (3 and 4, respectively in Figure 4.5a). Each media item also displays a name tag in the title bar (1 in Figure 4.5a), identifying the poster of the media item at a glance.

If someone is viewing a media item they do not own (i.e., they did not post it), they will (usually) only see the shrinking and hiding functions on the title bar; these allow the user to affect the item's appearance on their local NC board (Figure 4.5b). Finally, a media item can be moved around on the NC board by clicking on and dragging the item container's title bar.

The poster of a media item has exclusive control over removing it from the NC board. By default, disconnecting from the server will remove all the items a person has posted, which automatically 'cleans up' after them. While they are connected, a person can also remove any of their created items by clicking the '**X**' button on the media item's title bar (5 in Figure 4.5a). However, there will be times when a person may want their posted media item to persist beyond their login session. They do this by using the *timeout tool* (2 in Figure 4.5a), which allows the poster via a pop-up menu to specify how long a media item

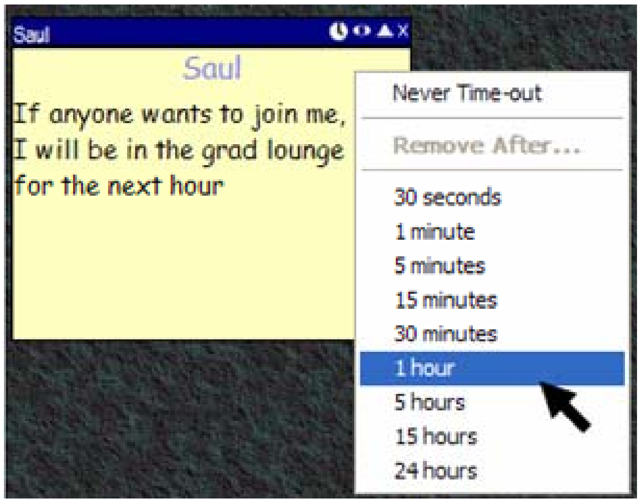
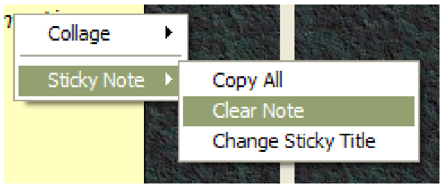


Figure 4.6 - A media item's timeout menu

should persist after its poster has logged out (Figure 4.6). Clicking this tool raises a menu with timeout choices ranging from one minute to twenty-four hours. When the time limit is set, the clock icon changes in colour on the owner's interface and appears on the viewer's (Figure 4.5c) to provide feedback for the new closing condition. The media item will automatically close on all connected NC clients when the time limit – tracked by the server – expires.

Media item containers also have context menus associated with them, where one can right-click on an item to gain access to further controls (Figure 4.7a). The hierarchical context menus have two sections: one labelled 'Collage' that gives the user access to NC board specific functions (e.g., the management controls also appearing on the title bars in Figure 4.5), and a second section with custom controls for the media item (as shown in Figure 4.7a).



A: Media item context menu

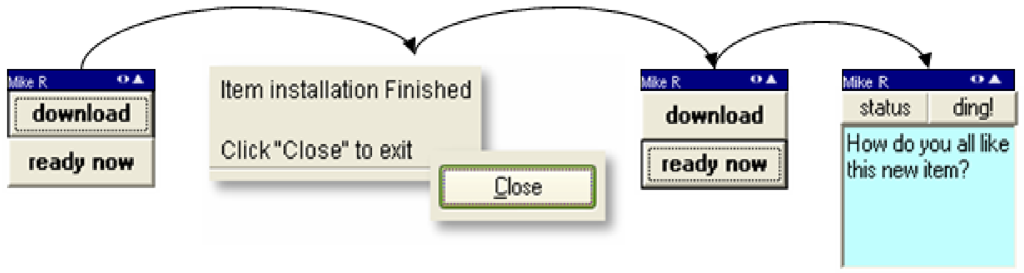
B: resizing media items

Figure 4.7 - Other media item features

Media item containers also have the ability to be resized. When the mouse is moved over an item, a resize button appears on the lower right-hand corner of the item (Figure 4.7b). This button can be clicked and dragged to resize the item. Not all media items will support resizing, but the button will always appear.

4.2.5 Dealing with New Media Items

As we will see in the next section, the Notification Collage is distributed with many stock media items that we believe are useful. However, we fully expect other developers to create new types of media items. Some may be generic items useful for many groups, while others may be specially designed for particular groups, people, and tasks. Thus one of the most important abilities of the new NC board is the ability to incorporate new media items on the



Item installation Finished

Click "Close" to exit

How do you all like this new item?

Figure 4.8a – prompt to download new item

Figure 4.8b – the user installs the new item outside the NC board client

Figure 4.8c – when install is complete, start new item

Figure 4.8c – the new item is immediately available

fly. While the next chapter describes how a programmer creates these new items, in this section we are concerned with how users install new items with minimal effort.

When a new item is made available, at least one community member must install it on their own local NC board. This is usually a fairly trivial exercise: the developer will likely post a standard Windows msi file (an installation file) on a web page known to the user, and the user needs only click on the link to begin installation. As in most windows installation, the user just needs to acknowledge the steps to be taken as presented in a series of dialog boxes. Once installed, the name of the new item will automatically appear on the 'Post Media' menu, and selecting it will create the new item on the local board (Figure 4.4). Note that this series of steps can be done as the Notification Collage is running – there is no need to shut down and restart the board.

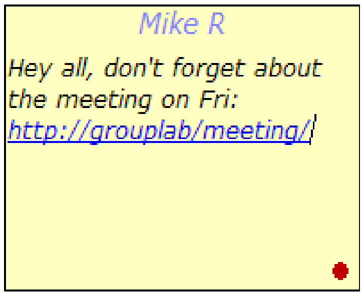
However, the problem is that at this point in time, only that person actually has a copy of the running software, and thus the Notification Collage boards owned by others cannot actually display the posted item on the NC board since the software has not been installed on their local system. Instead, other users see a special media item containing '**download**' and '**ready now**' buttons (Figure 4.8a). Pressing the 'download' button will tell the NC board to automatically navigate to the site containing the software and begin downloading the installation script for the new item. The users do not need to know where this site is, as all information is self contained by this special item. After completing the installation wizard for the new media item (Figure 4.8b), the user presses the 'ready now' button on the item (Figure 4.8c) to immediately use or join in on the current session with the new item (Figure 4.8d). These users can now create a new media item of this type through the 'Post Media' menu. Finally, this operation only has to be done once per new item, as the software is now included as part of the local Notification Collage. Again, all this can be done on the fly while the Notification Collage is running.

I must emphasise how this model significantly eases the task of updating the Notification Collage. As a distributed groupware system, the community membership may not be known, or people may come in and out of a session at various sporadic times. Anyone who has done software deployment for a distributed group knows what a nightmare this can be: software support to deliver up to date copies is tedious, and often requires a centralized authority. Even if software delivery succeeds, people will often neglect to install the software because of the work involved or because there is no perceived need to do so at that particular instance. Thus people often run out of date (perhaps incompatible) versions, and as a result one person's incorrect version can break the entire system. The deployment model of the NC changes this. Any person can install the first instance of the software, and others are notified of this new item as they are using the system. Installation is optional, and is just a matter of clicking a button.

4.3 The Media Item Stock Set

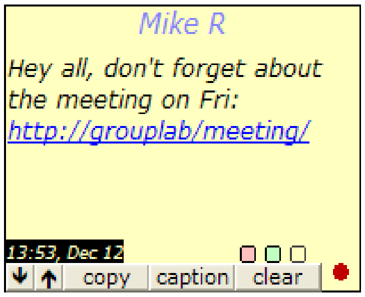
The initial set of media items presented in Chapter 3 demonstrated the ability for people to share and interact over different types of multimedia artefacts. Following these successes, the goal in rebuilding the media items was to maintain this level of interactivity and provide further opportunities for people to move into casual interaction quickly and smoothly. As media items are now added directly through the NC board interface, the media items themselves have been enhanced with their own controls to manipulate the multimedia information being shared (these are controls that would have previously existed on the posting clients). In this section, I first revisit the media items of chapter 3 and how they have evolved. I then present several new items designed to further support the bridge from informal awareness to casual interaction in a group of intimate collaborators, developed with help from fellow graduate students Kathryn Elliot and Michael Boyle. Note that while the surrounding media item container (e.g., the title bar and controls) is not shown in images in this section, it still appears with each media item on the NC board.

4.3.1 The Sticky Note


Mike R

Hey all, don't forget about the meeting on Fri: http://grouplab/meeting/

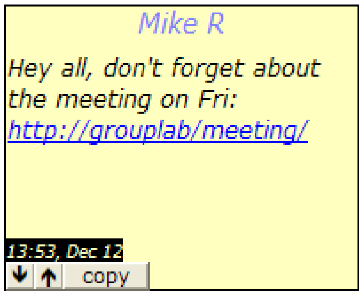
Figure 4.9a - *Sticky Note*, view on the NC


Mike R

Hey all, don't forget about the meeting on Fri: http://grouplab/meeting/

13:53, Dec 12

Figure 4.9b - Owner's controls on a *Sticky Note*


Mike R

Hey all, don't forget about the meeting on Fri: http://grouplab/meeting/

13:53, Dec 12

Figure 4.9c - A viewer's controls on a *Sticky Note*

The *Sticky Note* adds functionality to its previous version while still resembling a 'live' real world post-it note (Figure 4.9a). The poster of a sticky note can now type directly into their item with the NC. As before, edits are shown in real time on other peoples' NC boards in a mirrored sticky note item. The controls for the sticky note added to this version only appear when the mouse is dragged over the item. For the owner of the control (ownership is indicated by a red dot in the lower right-hand corner of the note), all controls are available (Figure 4.9b), while for people viewing the item, only the first three controls below are available (Figure 4.9c):

- a display of the time at which it was last updated;
- up and down arrow buttons will grow and shrink the font in the note, respectively;
- a 'copy' button copies the contents of the note to the Windows clipboard;
- three coloured buttons will change the background colour of the note to attract attention to it;
- a 'caption' button will allow the owner to change the caption on the sticky note (e.g., 'Mike R' in Figure 4.9), which is set to the name of the owner by default;
- a 'clear' button which clears the contents sticky note in one action.

The new sticky note also allows the owner to embed hyperlinks in the text they post. These links are automatically recognized as URLs and rendered as such when they are typed in (Figure 4.9). Clicking them from either view of the item will raise a web browser with the appropriate link followed. Finally, a scroll bar automatically appears when the contained text cannot all be displayed at the same time.

4.3.2 The Video Snapshot

The *Video Snapshot* item appears on the NC as a feed from a regular desktop web camera, just as before (Figure 4.10a). As with the *Sticky Note* (see the previous section), ownership of the video item is indicated with a red dot in the lower right-hand corner, and the owner (but not anyone else) can raise special control to adjust the properties of this item simply by passing the mouse over the item (Figure 4.10b). As well, the item contains a small caption in the upper left-hand corner. The particular controls let a user set the following attributes of their media item, which in turn affects how this item is displayed on all NC board.

- A pause button pauses the camera's capturing (paused video in Figure 4.10c);
- The time slider selects the picture taking interval;
- A button labelled with an 'A' allows the owner to change the video's caption;
- Adjusting the blur slider blurs the owner's image at different levels for privacy.

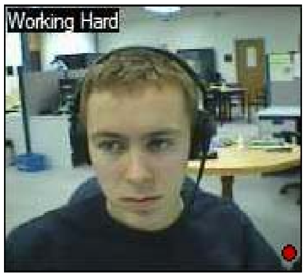


Figure 4.10a - *Video Snapshot* media item

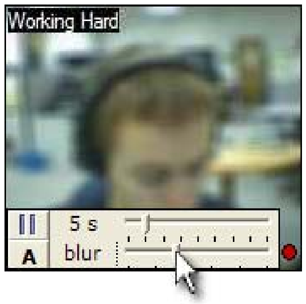


Figure 4.10b - *Video Snapshot* controls: adjusting blur

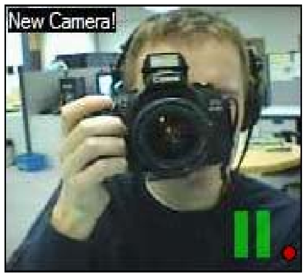


Figure 4.10c - *Video Snapshot* paused on particular image

Any time the owner clicks on the video snapshot, the camera will immediately take a picture. This feature works while the video is actively being captured and also while it is paused, letting the owner pick a specific scene to display (Figure 4.10c). In fact, all of the control tools work while the video is paused, allowing the owner to change their caption and the blur level on the video in the frozen state.

A natural extension of this media item is to let people click on the video to open a full audio/video connection. While this capability is not in the stock media item set, a later chapter will describe how this is included in an experimental version of this media item.

4.3.3 The Web Item

The *web item* allows people to share website links. Instead of using a custom web browser as before, the poster of this item now first raises the item's context menu to post a link (Figure 4.11a). After entering a URL and an optional description for the URL, the item goes to the web to capture a thumbnail of that page (Figure 4.11b). Once this is done, any person viewing the web item can click on it to raise a small window that contains a larger view of the thumbnail plus the optional description provided by the poster (Figure 4.11c).

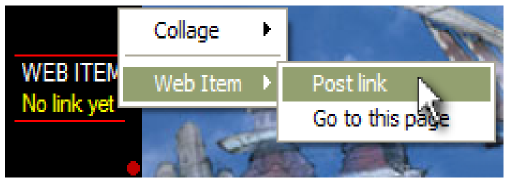
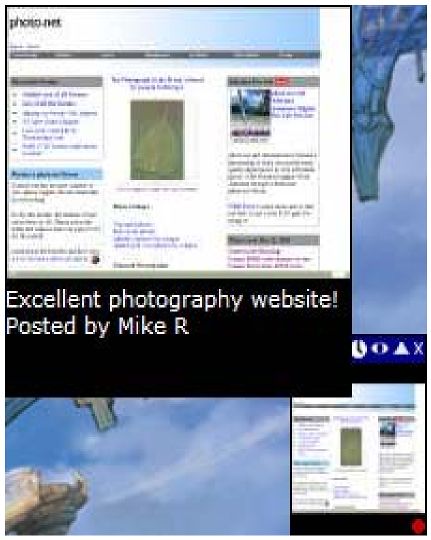
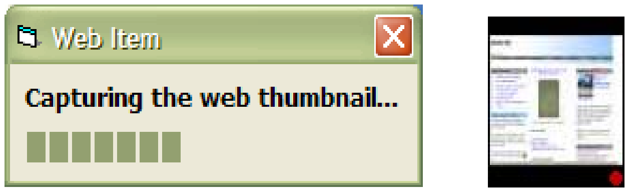


Figure 4.11a - Posting a link to a *Web Item*



Excellent photography website!
Posted by Mike R



Capturing the web thumbnail...

Figure 4.11b - After capturing a thumbnail of the desired link, a small copy appears in the *Web Item*

Figure 4.11c - Click on a *Web Item* to briefly view a larger thumbnail

As with the original web item, double clicking the thumbnail in the NC will open a web browser to the page represented by the thumbnail (not shown). Changing the link in the thumbnail simply involves repeating the above process with a new URL.

4.3.4 The Photo Show Item

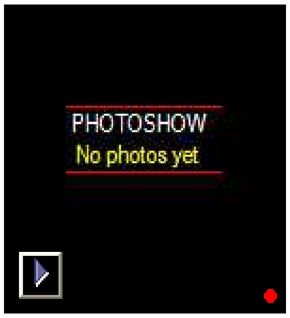


Figure 4.12a - The empty *photo show* item

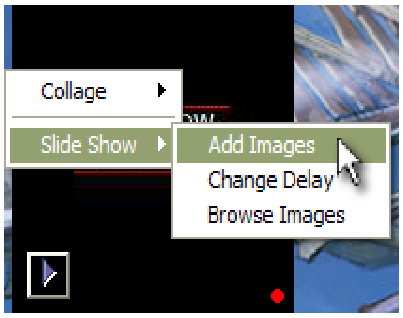


Figure 4.12b - *Photo show* menu includes posting and browsing

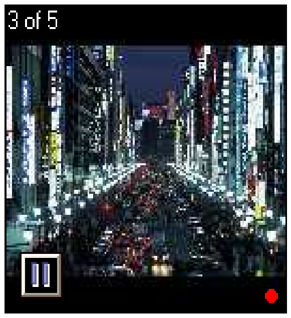


Figure 4.12c - Cycling through the set of images

The *Photo Show Item* has been updated significantly from the version designed for the initial NC board. When posted, the empty photo show starts in a paused state (Figure 4.12a). The owner adds one or more images from a file dialog box raised through the context menu on the photo show item (Figure 4.12b). One image is displayed at a time by the photo show item, but a count of the photos and the item's position in the show is displayed in the top left-hand corner of the item (Figure 4.12c). Pressing the play button in the lower left-hand corner of the item will make the item cycle through the photos, fading from one to the next image in the set every five seconds. Otherwise, simply clicking on the photo show item will advance it to the next picture in the series.

A new photo or set of photos can be added by the owner at anytime. Additionally, the interval between images can be changed through the context menu (Figure 4.12b). Finally, the photo show item allows anyone to download the full set of images at their full size by selecting the 'Browse Images' option in the item's context menu (Figure 4.12b). The images will be automatically displayed in the system's default image viewer.

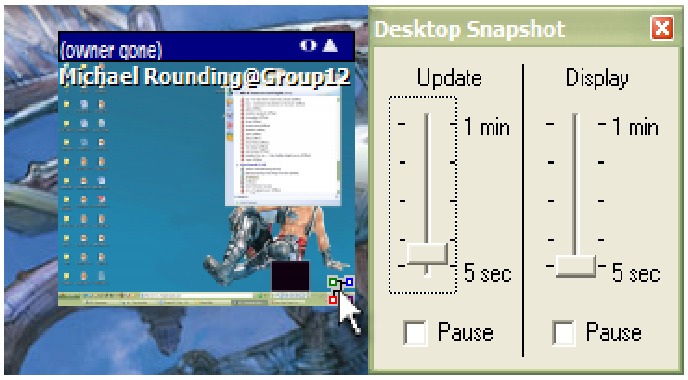


Figure 4.13 - *Desktop Snapshot* and its popup controls

4.3.5 The Desktop Snapshot

The new *desktop snapshot item* was developed by Michael Boyle but is in standard use with the NC board. As with the initial version, the intention is to be able to post snapshots of one's virtual desktop. The item is now a public item however, which means that everyone can choose whether or not they wish to participate in sharing their desktops. The item itself shows one person's desktop at a time, along with their name and the machine the snapshot is coming from (Figure 4.13, left). Should another person wish to join by posting their desktop, they click on the small icon in the lower right-hand corner of the item to bring up its control panel (Figure 4.13, right). By default, for privacy reasons everyone's 'Update' is paused. A person has to explicitly unselect the 'Pause' check box at the bottom of the column, to add their desktop to the list, and a new snapshot is taken on the interval defined by the slider in the same column.

The display of desktop snapshots is independent from their capture, i.e. they are timed independently. Adjusting the controls on the right side of the control panel changes the interval between peoples' desktops in the item, or pauses the playback completely. Since the item exists on the NC board without an owner, anyone can actually remove the item through its context menu.

This current version only displays the desktop in miniature, and is just a static image. This does not follow the NC philosophy of letting people act on an item to move into interaction. As we will see in a later chapter, we have also developed an experimental version of this media item that allows people to actually connect to another's desktop and interact with that person.

4.3.6 NC Availability Item

The NC *availability item*, developed by Kathryn Elliot, is designed after the iconic status indicators present in instant messaging systems (item shown in Figure 4.14a). When people cannot post video, they can use this item to give others some idea of their availability. By default, the six different availability states a person can choose from the item's context menu are similar to the generic states provided by IM systems (for example 'online', 'away', or 'be right back'; Figure 4.14b). Changing from one state to another also changes the image displayed by the item. The *availability item* can be customized by choosing the 'change my images' option from the context menu. This brings up a dialog where the captions and pictures for availability states can be changed (Figure 4.14c). For

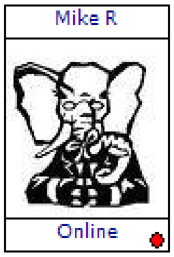


Figure 4.14a - the *Availability item* as it appears on NC

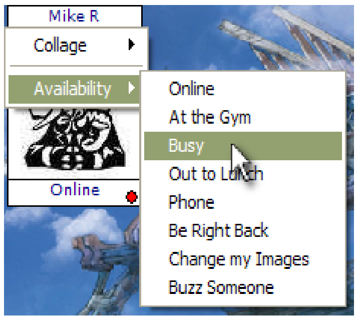


Figure 4.14b: the *Availability item*'s context menu controls

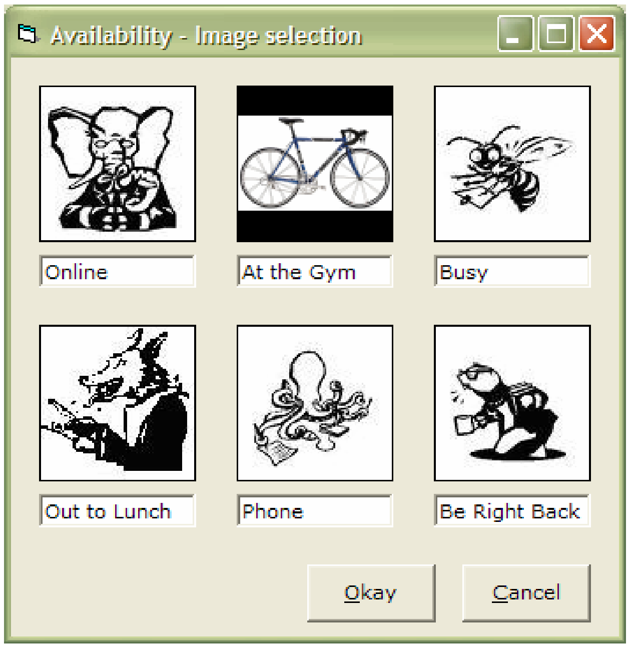
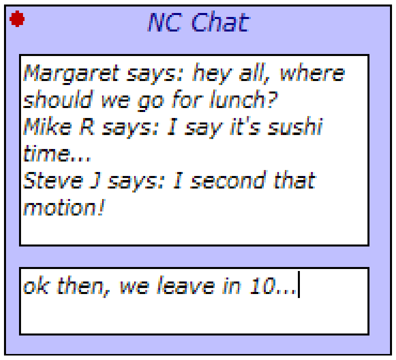


Figure 4.14c: customizing the *Availability item*'s pictures and status messages

example, people typically use photos of themselves or caricatures to indicate (often whimsical) versions of themselves in different states.

4.3.7 The NC Chat Item



Margaret says: hey all, where should we go for lunch?
Mike R says: I say it's sushi time...
Steve J says: I second that motion!

ok then, we leave in 10...

Figure 4.15 - the *NC chat* item

The NC *chat item* is a text item also developed by Kathryn Elliot (Figure 4.15). Its need came from the observation that users of the initial NC had a tendency to clutter the display with many *sticky notes* when large conversations were occurring. Every speaker had one or more *sticky notes* up, and each one competed for space and attention. While conversational threads could be followed easily by attending the unfolding talk, new participants would have to interpret all these seemingly diverse text fragments. As well, there is no persistence: if text is erased or a media item deleted, that part of the conversational record disappears.

In contrast, the chat item is more similar to chat systems that puts one conversational thread into a single display, where it allows several people (not only its owner) to contribute to the conversation at once via the lower pane. Each contribution, identified by the user-name they signed in with, is added to the bottom of the upper pane in time order. Its interface is quite simple: the only controls appearing in the context menu are to clear the chat buffer and change the caption on the chat window. Scroll bars automatically appear when needed.

4.3.8 The NC Sketchpic Item

Recognizing that people should be able to move smoothly from awareness to full collaboration and back again, we decided to create a media item that has different granularities of operation: a small form for awareness, and a large form for actual work. Our test bed was a groupware sketchpad that operated over images. To this end, the *NC Sketchpic item* takes inspiration from student projects by Kathryn Elliot and Rosemary Sanchez created as part of assignments to build new media items (see chapter 5). The version included with the NC posts a picture of a sketching session as a thumbnail to the NC board (Figure 4.16a). A list of names on the thumbnail shows who is actively participating in the sketching session. As the sketch evolves, the thumbnail is continually updated to show others connected to the NC board the current state of the sketch, albeit in miniature and with no telepointer displayed. The thumbnail also updates to show when new people join in on the sketching session (Figure 4.16b).

To join the sketching session, a person double-clicks on the Sketchpic item. This brings up a full size sketching canvas window, which is displayed independent of the NC (Figure 4.16c). The sketching canvas is a groupware sketching interface with a number of controls outlined in Figure 4.16c. Sketchpic allows people to sketch over a blank canvas or various images; images the user can select through Sketchpic controls include photos or diagrams stored in image files, a desktop snapshot of a person's virtual workspace, or a single window on a person's desktop. These are loaded by the controls numbered 1, 2 and 3 in Figure 4.16c, respectively. Emptying the canvas (5 in Figure 4.16c) will allow people to sketch over an empty canvas, while clearing the sketch (6 in Figure 4.16c) will clear annotations added by people but not the image loaded as the canvas background. Should someone want to save the results of a sketching session to an image file at any time, they do so by clicking on the 'save sketch' button (4 in Figure 4.16c). People can control the look of their annotations by changing the colour and width of their paintbrush (7 and 8 in Figure 4.16c, respectively). All of these controls apply to the sketching canvas (10 in Figure 4.16c). People can gesture to others on the canvas using tele-pointers, which are activated on a per-user basis by right-clicking on the canvas and moving the mouse around (11 in Figure 4.16c).

The sketching canvas also provides a chatting facility similar to the NC chat item, allowing people to include textual messages along with their annotations (12 in Figure 4.16c). People participating in the sketch are tracked by a user's list, each colour-coded with that person's annotation colour (9 in Figure 4.16c).

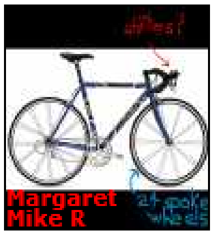


Figure 4.16a - the *NC Sketchpic* Item with a list of people participating

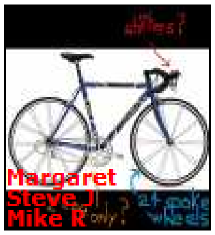
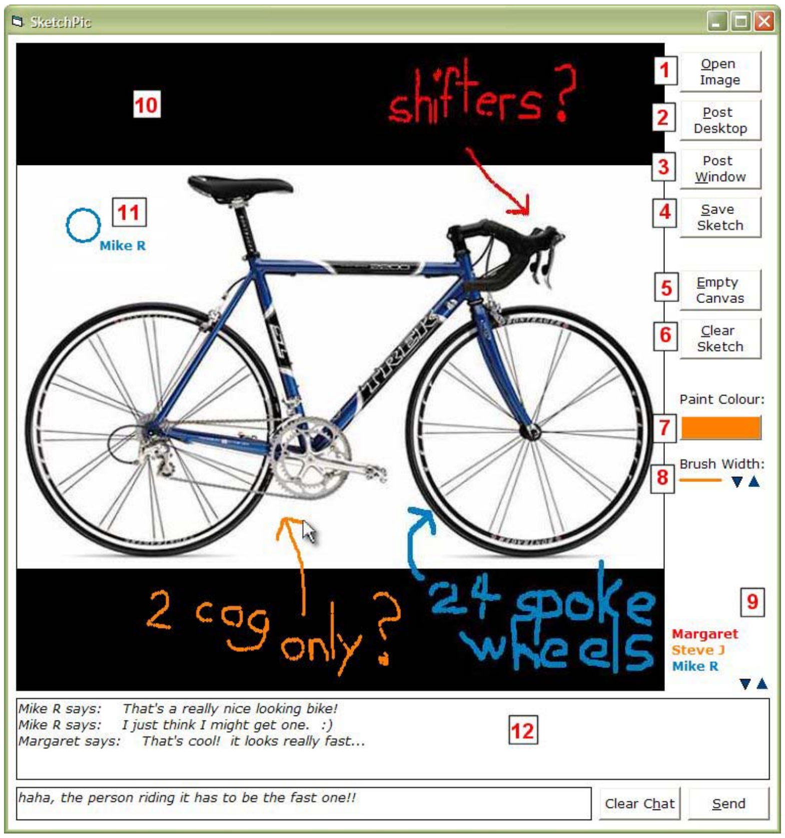


Figure 4.16b: *NC Sketchpic Item* updating as people draw or join in



Mike R says: That's a really nice looking bike!
Mike R says: I just think I might get one. :)
Margaret says: That's cool! it looks really fast...

haha, the person riding it has to be the fast one!!

Figure 4.16c: the *NC Sketchpic Item's* sketching canvas window controls

1. Open Image button
2. Post Desktop button
3. Post Window button
4. Save Sketch button
5. Empty Canvas button
6. Clear Sketch button
7. Paint colour selector
8. Brush Width selector
9. Users list
10. Sketch Canvas
11. Tele-pointers
12. Chat interface

4.4 Summary

This chapter explained the redesigned NC board user experience. I have described the new NC board from the user's point of view. I paid particular attention to how the NC was re-implemented as a single user interface that allows people to connect to and use the public space. Media items are now created directly within the NC board, and their contents are manipulated in place. Finally, I described the stock set of media items, which now provides for richer control and interaction over multimedia information and more importantly that demonstrates how one moves from awareness to interaction and work. I also described how new media items can be included.

In the next chapter, I focus on the technical aspects of this re-implementation and how it is built. These aspects are described as several components which are consolidated as the *media item architecture.* As mentioned at the end of the previous chapter, development of both the architecture and the new interface were co-supporting objectives, and it is important to remember that they occurred in parallel.

Chapter 5. The Media Item Architecture

In chapter 3, I introduced the Notification Collage (NC) and our initial development and usage experiences with it. What came out of those initial experiences were several recommendations leading to a redesign, with the most important idea being to merge all the various clients – the Notification Collage board, the various media item clients that individuals use to capture and present information – into a single, unified client interface. This interface was presented from a user's point of view along with a stock set of media items in chapter 4.

The redesign idea had major architectural implications. In particular, we would expect third-party developers to create media items that would work within this unified interface. However, it would be unrealistic – from both a development and a deployment point of view – to give developers NC source code access. Modifying the source would be too difficult, and trying to manage and re-distribute different versions of the source to the community would be hard to do in practice (especially if incompatibility issues were to appear between versions).

Consequently, I developed a new architecture to mitigate these problems along with the new NC board interface. In this chapter, I present the *Media Item Architecture*, a software and data architecture designed to separate the development of the NC board interface from the development and deployment of media items. I do this first by forming a conceptual model for redesigning the NC and this architecture. That model is then used to derive four goals, each addressing recommendations found in chapter 3. In the remainder of the chapter, I present a new architecture that explains the construction of the NC board presented in chapter 4 by: incorporating a more robust and flexible shared dictionary system, including a scheme where media items can be loaded into the NC on the fly and forming an input / output paradigm for media items appearing on NC. Because the architecture is fairly technical, I introduce it at several levels, with each level showing progressively more detail. I close by discussing this development model and how it allows rapid prototyping of new media items from experiences with an undergraduate computer science interface design course.

5.1 Conceptual Model, Architecture Goals and Overview

The new architecture is formulated around a new conceptual model of how the NC should work.

1. The *NC board* (Figure 5.1, top) is a client that hosts *media items*. It positions items within an interface similar in appearance to the original NC, and it lets people move around and control these items in generic ways. However, it knows nothing about the content of media items i.e., it is not responsible for what is displayed within the media item, or how the media item should respond to user input. In this way, it is similar to a window manager that knows about windows on a screen but not about their contents.

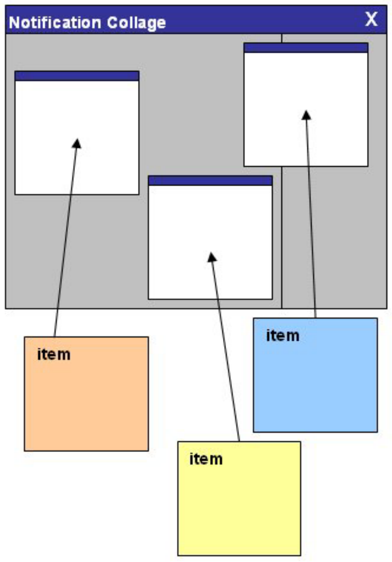


Figure 5.1 - Conceptual Model

2. *Media items* (Figure 5.1, bottom) are discrete, programmer-defined visual objects. They display output, and they understand user input. A particular media item shares state information with their partner instances on other displays. However, media items do not exist by themselves. Rather, they must be 'hosted' by an application (e.g., the NC board), which positions them on the screen so the user can interact with them.

3. *Dynamic hosting* (Figure 5.1, arrows from items to generic containers) means that the NC board has a well specified way of incorporating media items at run time, regardless of whether these items have been seen before.

This conceptual model of the new NC means that developers can create media items independent of the NC board, that they can deploy them without any recompilation or redistribution of the hosting NC, and that the people using the board will be able to use previously unseen items without having to shut down and restart the NC board application.

5.1.1 Architecture Goals

The new NC architecture derives mainly from the conceptual model above, as well as other recommendations in chapter 3. Its specific implementation goals are detailed below.

1. **Use a robust shared dictionary that lets the Notification Collage distribute, store and control both simple data structures and multimedia across the internet.** The primary purpose of the shared dictionary is to provide a single data structure that lets NC board instances as well as media item instances to independently share information. Since I am designing from scratch, I can completely replace the shared dictionary system previously used with a new one. As we will see, the new shared dictionary system is used to specify a hierarchy that separates media information (as used by media items) from user information (as used by the NC board). It is more flexible and robust, and allows a reliable client / server communication paradigm over the Internet. It also allows a developer to control persistence, where media items can be left up for a time duration independent of its poster's connection status.

2. **Media items should be dynamically added to the NC board without requiring recompilation or re-installation.** I devise a scheme where media items can be loaded into the system on the fly. If new items obey an agreed-upon object interface, they will be hosted by the NC board. This will allow the board to host individual media items without knowing their contents. It will also allow new items to be incorporated dynamically without interrupting the operation of the NC board or other media items at runtime. This will eliminate the bulky upgrading process of the initial NC (recompilation, deployment, installation, etc.). This is important if third party developers are to deploy media items they have created.

3. **A media item should serve as both input and output.** The new generation of media items will both capture local information to broadcast to others, and will display information received from their remote counterparts. This does away with the many separate client-side interfaces of the previous NC (e.g., as shown in Figure 3.3). The modality of this operation (whether input, output, or both) will be determined at run-time.

4. **Media items should provide a simple programming model so that average programmers can rapidly prototype new media items.** The architecture will serve double duty as a toolkit for rapid development and deployment of media items, as well as a groupware interface. To evaluate whether the programming model is indeed easy to use, we will test how programmers explore novel designs for media items.

5.1.2 Overview of the Architecture

This section provides an overview of the new *Media Item Architecture.* It adds to the conceptual model presented earlier, where it details how media items interoperate with item containers which in turn are hosted by the NC board, showing how the interface presented

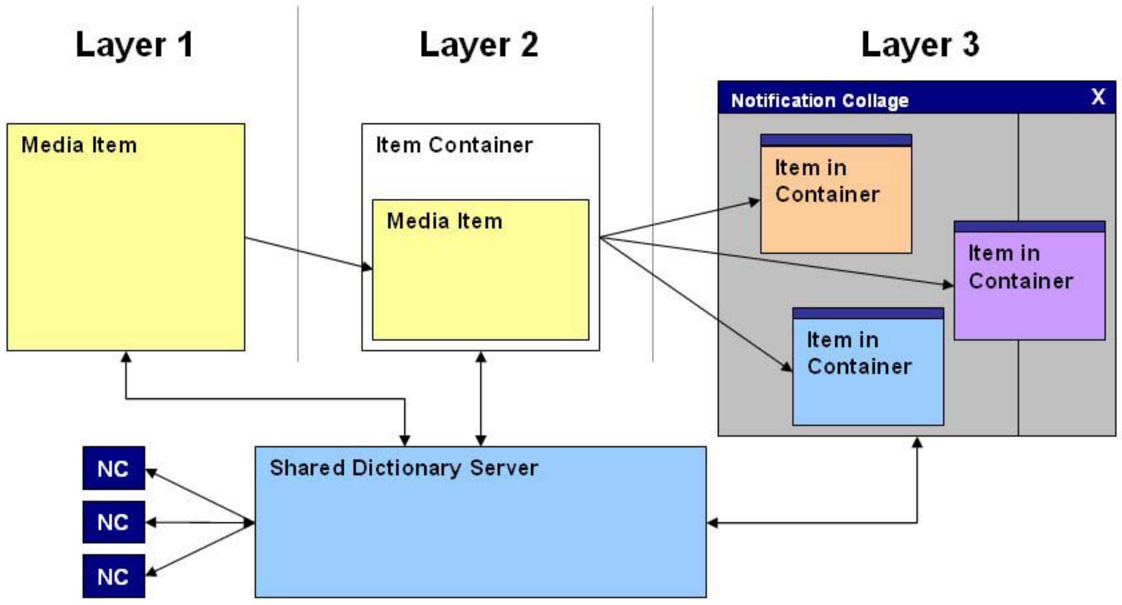


Figure 5.2 - Design for the Media Item Architecture

in chapter 4 was constructed. Figure 5.2 illustrates the architecture in three layers working with a shared dictionary server:

1. **Layer 1**: Each media item is built as a distinct object, separate from the NC board (layer 3) and completely responsible for capturing any user input and displaying its output. It is also responsible for distributing shared data to its instance counterparts through the shared dictionary connection. It implements a software interface recognized by a media item container class (layer 2).
2. **Layer 2**: Each media item is 'hosted' by a generic container class, which resembles a small window. As we will see, the NC board knows how to dynamically load and host these containers. Thus a container acts as a transparent proxy that separates a media item (layer 1) from the NC board (layer 3), while still allowing the two to interoperate. While the NC board knows how to deal with these generic containers, it need know nothing about media items they contain.
3. **Layer** 3: The Notification Collage board, described in chapter 4, is responsible for hosting multiple containers, where each container (layer 2) is capable of containing a separate media item (layer 1). Each time a new item needs to be added, the NC board loads a new container. The container, which actually loads the item, provides a layer of abstraction between media items and the NC board, which in turn allows new items to be added dynamically without recompilation or reinstallation.

The main hub of information exchange in this redesigned architecture remains a client/server based shared dictionary (SD) system (bottom of Figure 5.2). Goal #1 as presented in the previous section is to use a more flexible and robust SD system. This is accomplished by adopting the SD component of the Collabrary, a toolkit for developing multimedia groupware (Boyle & Greenberg, 2002). The NC board will be responsible for establishing and maintaining the SD connection, which will be shared across all three layers. This means that each media item on the NC board will use the same network connection as the NC itself.

After first explaining how the architecture will capitalize on the newly adopted SD system, I will describe how the three layers of the architecture combine to accomplish the goals of this redesign. I will describe the object interface that media items use followed by discussing the hosting mechanism. I will then present the development model for building new media items. Finally, I will show the results of distributing the development model to a group of average programmers to explore novel designs for media items.

5.2 Collabrary: A Robust Shared Dictionary for the NC

From our experiences of the previous NC, a shared dictionary (SD) client / server proved a successful general purpose data sharing and notification server, where it was an elegant means to represent, transmit and store information. Unfortunately, the initial shared dictionary server was unreliable in operation and led to frequent and unexpected service outages, sometimes with multiple crashes in a single day. Since the new NC would be rebuilt from scratch, I could completely replace the unreliable SD with a newer more robust and powerful version. Consequently, I used the shared dictionary component of the Collabrary, a toolkit written by Michael Boyle designed to enable rapid-prototyping of multimedia groupware (Boyle and Greenberg, 2002). While the Collabrary was built as a general toolkit, the new NC was its first major test bed; so consequently the Collabrary shared dictionary and the NC evolved in tandem.

While the Collabrary shared dictionary (SD) is similar in concept to the server component of the shared dictionary described in section 3.1.1, it is very different in implementation. However, it too is conceptually like a notification server (Patterson et al, 1996) augmented to cache the most recent data sent to it in a central repository. The SD server maintains an active list of clients who are connected to it, as well as information that a programmer can use to control how data persists in its repository.

This section is not an exhaustive description of the shared dictionary's features, but is intended to explain the concepts crucial to the development and implementation of the media item architecture and the unified NC client. While I am not principle developer of the Collabrary, the Notification Collage was the primary driving force behind the Collabrary shared dictionary and I contributed to its intellectual formation. Sections 5.2.1 – 5.2.3 provide a brief tutorial to the Collabrary shared dictionary. Section 5.2.4 describes the Notification Collage's SD-based data model.

5.2.1 Connecting and Instances

Figure 5.3 contains an excerpt of Visual Basic 6 (VB6) code that demonstrates how an SD client instance is created, how a connection to the server is opened, and how the programmer detects a successful connection.

After declaring the SD as a `Collabrary.SharedDictionary` instance and creating that instance in the `OpenConnection` procedure, the `sd.Open` call actually opens the client[1]. The argument, in URL syntax, specifies the server host machine name (`example.machine.com`) and the name of the server process ("`test`"). When a connection is successfully opened, the SD client fires an event which in turns invokes the callback function `sd_Opened`. In this example, the callback merely displays a message box but of course other actions can be taken e.g., initialization actions. The client can also fire other relevant events (e.g., user exited or connection closed) to update the programmer on other status changes to the SD connection. In terms of the new NC board, the connection dialog

```
Private WithEvents sd As Collabrary.SharedDictionary

Private Sub OpenConnection()
     Set sd = New Collabrary.SharedDictionary
     sd.Open "tcp://example.machine.com:test"
End Sub

Private Sub sd_Opened(ByVal Url As String, ByVal isServer As Boolean)
     MsgBox sd.Me
End Sub
```

Figure 5.3 – SD client setup code

[1] Since the NC is developed as a client application, only the client form will be presented here. However, the Collabrary includes simple tools to start and administer SD servers, and these default servers are used by the NC.

(Figure 4.2) gets this server location and name information in its ‘Server Info’ section.

Internally, the SD maintains an active list of users connected to the server. The SD refers to each connected user as an *instance.* Each instance (and thus each user) is identified by an automatically generated globally unique identifier (GUID) – a 128-bit value represented as a character string enclosed in braces. Individual clients can access their own GUIDs by using the “Me” property of the client. For example, the `MsgBox sd.Me` call in Figure 5.3 will display the local instance’s (user) GUID in a message box after a successful open.

5.2.2 Shared Dictionary Keys

Keys in the SD are hierarchical strings specified as paths. The hierarchy is specified by using the solidus (“/”) character to separate different levels of the hierarchy in a specific key. For example, Figure 5.4 shows keys in a hierarchy foreshadowing the way the Notification Collage actually stores its information. Users are specified by the root key “`/users`”, and media types by “`/media`”. Individual users and media items are defined by child keys (e.g., “`/users/user1`”, “`/media/postit1`” and “`media/video1`”). Under each

SD Keys	Values Stored at Keys
/media/postit1	<unique id #>
/media/postit1/message	“Hi there!”
/media/postit1/caption	“Mike R says”
/media/postit1/.transient	“user1”
/media/postit2	<unique id #>
/media/postit2/message	“Hi, what’s up now?”
/media/postit2/caption	“Sally B says”
/media/postit2/.lifetime	60
/media/video	<unique id #>
/media/video/snapshot	<serialized video frame>
/users/user1/name	“Mike R”
/users/user1/url	“http://www.azedzed.net/”
/users/user1/email	“rounding@cpsc.ucalgary.ca”
/users/user2/name	“Sally B”
/users/user2/url	“http://sally.com/”
/users/user2/email	“sally@someplace.com”

Figure 5.4 - Stylized NC SD Hierarchy

of these are further keys defining information for individuals and media items e.g., “`/media/postit1/message`” for a message, “`/users/user1/name`” for a person’s name, “`/media/video/snapshot`” for a video frame, and so on.

SD keys can hold a value, and these key / value pairs are what makes a dictionary structure. That is, SD keys are both place-markers in the hierarchy and spaces to hold information as well.

The programmer's interface is similar to that of standard dictionary data structures (a.k.a., hash table, map, associative array) in that the programmer specifies a key / value pair. The key is the index into the hierarchy, and the value is what should be stored there. Values stored in the SD can range from simple strings to complex objects or binary information marshalled for storage. This ability to incorporate complex and binary objects is a significant improvement over the older dictionary implementation as it makes multimedia programming less complicated. Removing keys from the SD is as simple as specifying an empty value to be stored at the desired key.

To illustrate, Figure 5.5 shows an excerpt of code written in VB6 that demonstrates creating and storing a key/value pair (`sd(”/media/postit1/message”)=“Hi There!”`), retrieving a value (`MsgBox sd(“/media/postit/message”)`), and then removing a key/value pair from a SD (`sd(“/media/postit1/message”) = Nothing`).

```
Private Sub SDTest()
    sd("/media/postit1/message") = "Hi there!"
    MsgBox sd("/media/postit1/message")
    sd("/media/postit1/message") = Nothing
End Sub
```

Figure 5.5 – storing and retrieving using Shared Dictionary

Ordinarily, keys stored on the SD server persist indefinitely and independent of any instance connection. However, the SD specifies two highly useful reserved key names: ‘.transient’ and ‘.lifetime’. Both are used to control the lifetime of sub-trees in the SD hierarchy:

- The '.transient' key makes data persist only for the lifetime of the connected client. Its value is loaded with the user's GUID representing the client (see the previous section). When that client is disconnected, the key one level up from the .transient key plus all its children (including the .transient key) are deleted. To illustrate, if `'user1'` in Figure 5.4 is disconnected, the shared dictionary automatically notices that user1's GUID matches the .transient value in the `/media/postit1` hierarchy, and that entire sub-tree would be removed.
- The '.lifetime' key allows a developer to force data to persist for a specified time duration. Its value is loaded with a time value (in seconds). The shared dictionary server (not the clients) tracks this time from when the value is set, and will remove its parent and all its children when the specified duration has passed. In Figure 5.4 for example, the `/media/postit2` hierarchy will be removed automatically after a duration of sixty seconds, as specified by the `/media/postit2/.lifetime` key in that hierarchy.

By strategically placing '.transient' and '.lifetime' keys in the SD, the new NC board can meet part of architecture goal #1 stated earlier, where media items can be either tied to the connectivity of the poster or remain on the NC independent of the poster's login status.

5.2.3 Key Subscription

When keys are created, changed or deleted, especially from a different client, the local client needs to know about it. The SD does this through a *subscription mechanism*, where a programmer can selectively attend to particular keys.

In particular, programmers can *subscribe* to keys or patterns of keys, and associate with them a programmatic call-back that is invoked whenever keys matching a subscription pattern are modified. Modifications to keys are detected in one of four ways:

- **Added** – A key is appearing in the SD for the first time, and a value associated with it is being cached by the SD server.
- **Changed** – The value of a previously existing key has been modified.

- **Removed** – A previously existing key and its value have been removed. This key will no longer be available for any client to read.
- **Signalled** – A key has been used to broadcast out a value of some kind, but the value will not be cached. This implements a pure notification server (Patterson et al, 1996), where the server only broadcasts, but does not store information.

To illustrate, Figure 5.6 shows a code snippet that subscribes to the key "`/media/postit1/message`". First, the programmer creates a subscription object (provided by the Collabrary toolkit). When the programmer is told that the SD is successfully opened, they initialize the subscription object to either a single key or a pattern of keys to watch for. When one of the above events occurs on a key matching the subscribed pattern, the subscription object will receive the appropriate event notification as well as the reason for the notification (e.g., added, changed, removed or signalled) so it can be dealt with on an individual basis. In Figure 5.6, for example, if any client anywhere adds or changes the key defining postit1's message (introduced in the hierarchy of Figure 5.4), the `sp_Notified` callback is invoked, and new text is displayed in a message box.

The SD also uses pattern matching through a regular expression system that enables a programmer to subscribe to all keys matching a given pattern. For example, through the 'wildcard' character ('*'), programmers can subscribe to an entire hierarchy of keys and receive notification of changes that happen to any key in it. To illustrate, if the subscription

```
Dim WithEvents sp As Collabrary.Subscription

Private Sub sd_Opened(ByVal Url As String, ByVal isServer As Boolean)

     Set sp = sd.Subscribe("/media/postit1/message")

End Sub


Private Sub sp_Notified(ByVal Key As String, ByVal instance As _
     String, ByVal val As Variant, ByVal reason As _
     SubscriptionNotifyStyle, ByVal prev As Variant)

     If reason = Added OR reason = Changed Then _
        MsgBox val
End Sub
```

Figure 5.6 – Subscription to a key pattern

in Figure 5.6 was to "`/media/postit*/message`", the callback would be invoked when any postit's message key was changed.

5.2.4 NC Data Model

Similar to the preliminary NC, the new NC be designed around a model-view-controller data-sharing architecture (Stefik et al, 1986). As such, a good design for the data model using the new SD system is needed.

The data model will contain two basic types of information. As with the previous NC, it will contain information about media items. However, it will now also contain information about connected individual users so that data about those users can be exploited by the NC board e.g., to show who is logged on and whether or not those people have posted media items. These two information types will be maintained separately. Indeed, a large advantage of adopting the Collabrary Shared Dictionary is the ability to separate information via the hierarchy.

The user information and media item data model will be described independently in this section. It is important to note though, that they are interrelated as *people* post *media items* and there does need to be a way to get from one to the other.

5.2.4.1 The 'Users' Data Hierarchy

When connecting with an NC client, a user supplies some basic information about him or herself (name, email address and homepage URL are examples, see section 4.2.2). This information is stored on the server using the data model, and all NC clients can access this information. For example, the NC Board uses this information to manage and display its visible 'online' list, which in turn notifies people about who is currently connected to the NC system and allows them to gain further information about them (see section 4.2.3).

All user information appears on a per-instance basis under the '/users' key hierarchy of the SD. Each user instance is assigned a unique identifying key `'/users/<instance GUID>'`; its value stores a Collabrary structure[2] storing the following information:

- **name** – the name of the person, either as a short or full name.
- **email** – the email address of the person (optional).
- **homepage** – the URL of the person's homepage (optional).
- **phone** – the person's phone number (optional).
- **caption** – a caption to be used by some media items (optional – if not used, the person's name will be used instead).

5.2.4.2 The 'Media' Data Hierarchy

Information pertaining to media items will be stored in a separate branch of the shared dictionary hierarchy, all keys prefixed with '/media'. As with the user hierarchy, each new media item is stored on a per-instance basis by first adding a unique identifying key as a child of /media i.e., `'/media/<instance GUID>'.`

When the user selects a media item type to post from a menu on the NC board (see section 4.2.4), it first it creates a new GUID for that item. Having a different GUID, or id, for each item means that more than one instance of the same media item type can appear on the board – this is a more robust version of the id numbering scheme described in chapter 3. Using information held in the Windows registry file, the NC then dynamically loads the dll containing the media item's code. After the item is initialised, the NC board passes control to the media item itself, which is responsible for posting the following generic Collabrary structure at the `'/media/<new GUID>'` key:

[2] The Collabrary toolkit provides a generic structure object that allows the developer to store a series of data fields indexed by string-based names. The object is automatically serialized when stored in the SD.

- **DownloadUrl** – a string describing a URL where an installation script for the media item can be found. If someone connected to NC is missing a particular media item, this value is used to retrieve and install media item implementations on the fly.
- **MIMEType** – a string that describes the primary data type of the information presented in the media item. For example, the *Sticky Note* as described in the previous chapter has a MIME (Multipurpose Internet Mail Extensions) type of 'text/ascii', and a *Photo Show* has 'image/jpeg. The SD can store binary data, but often this data is merely a stream of bytes: the **MIMEType** value makes it possible for other researchers to build logging tools (Tang & Greenberg, 2002) that can fully understand media item data without needing to know specific details about the media item implementation.
- **RootPath** – a string representing the key at which this information structure has been stored, thus its value will always be `'/media/<new GUID>'`.
- **RelativePath** – a path that, when appended to **RootPath,** will find the primary information displayed in the media item. This is the information whose data type is described by the **MIMEType** field. For example, a *Sticky Note* stores its message contents at a key under its root called `'/media/<new GUID>/message'`, so the **RelativePath** is `/message'.` As with the **MIMEType** value, this value is intended to provide future support for generic logging applications (Tang & Greenberg, 2002).
- **ProgID** (Program Identifier) – the name of the media item as installed on the system and registered with the operating system registry. This information is used by other NC clients to load the media item described by this structure. The value stored here is the same as the value stored in the operating system registry file, mentioned earlier.
- **Owner** – the GUID that matches the instance GUID of the person who posted the media item. This is used to map into the Users data hierarchy as described in the previous section, where one can find information about the person who posted the media item.

Every media item is required to post the above generic information structure to the SD server, and adhere to the conventions for storing information in it. Every time one of these structures is added to the SD data model, the server will signal all connected NC clients (through the Model-View-Controller paradigm) that a new media item has been added. Similarly, when the structure is removed, the data model signals the item's removal. Using this arrangement, there will be enough information for an NC client to either load the appropriate media item, or to find a place to retrieve it from on the fly. Media items are free to store any additional information they require under the `'/media/<new GUID>'` key (for example, the *Sticky Note*'s message: `'/media/<new GUID>/message'`). This arrangement effectively partitions the `'/media'` hierarchy into isolated sub-trees, one for each posted media item.

5.3 The Media Item Library

The previous section detailed how the new shared dictionary will be a paradigm for data sharing, which satisfies the first goal in Section 5.1.1. This section addresses the second and third architecture design goals by showing how the *media item library* provides a method to dynamically load media items into the NC, where the items support both input and output to streamline the user experience. I describe how this software library fits into the different layers of the architecture described in section 5.1.2 to build the new NC board client described in chapter 4. As part of the scheme for dynamically loading media items, I also introduce the development model used to create media items and provide a test-bed to solve the fourth design goal.

5.3.1 Library Overview

Figure 5.7 illustrates the overall design of the media item library and how it fits into the media item architecture. Figure 5.7a describes the different objects available in the library, and Figure 5.7b adds implementation detail to the architecture diagram presented in section 5.1.2, placing the crucial parts of the library in the three different layers.

Recall that a primary goal of the architecture is to separate media items as much as possible from NC board implementation, which in turn enables independent development and deployment of media items and the NC while still allowing them to interoperate. This is done by using the shared dictionary data model (section 5.2) and the generic software interface described next.

Layer 1 (see Figure 5.7b) specifies that each media item is implemented as an ActiveX control. This is a Microsoft facility that lets a programmer create a visual component (e.g., a widget) and distribute it as an independent dll. ActiveX controls can then be included as separate objects by other software systems. In this case, special ActiveX controls (media items) will be included as objects by the NC board.

Library Component	**Function**
IMediaItem interface	A software object interface defined in the library and implemented by ActiveX controls to become media items.
MediaItemHost control	An ActiveX control that hosts media items, or other ActiveX controls that implement the *IMediaItem* interface.
MediaItemEventObject object	An object that lets media items forward user interaction events to be synchronized on through the different layers of the architecture
ShortcutMenu object	An object that uses a shared dictionary hierarchy to describe a menu hierarchy. It allows media items to specify and export a context menu definition.

Figure 5.7a – Object contents of the media item library

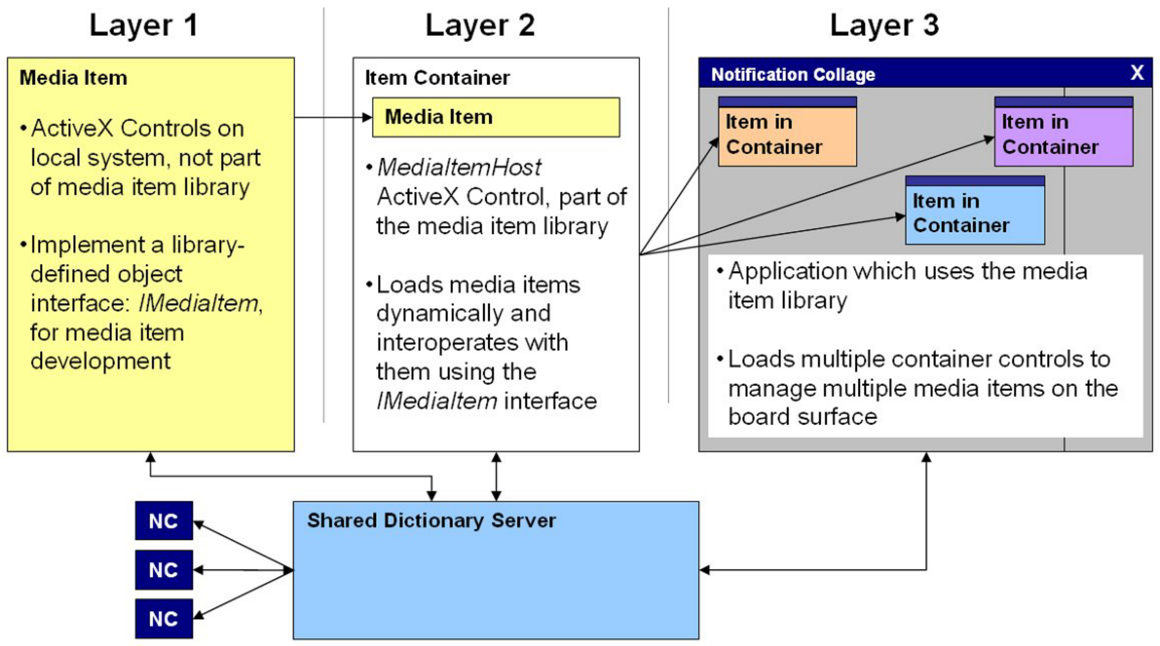

Layer 1
Layer 2
Layer 3
Media Item
• ActiveX Controls on local system, not part of media item library
• Implement a library-defined object interface: IMediaItem, for media item development
Item Container
Media Item
• MediaItemHost ActiveX Control, part of the media item library
• Loads media items dynamically and interoperates with them using the IMediaItem interface
Notification Collage
X
Item in Container
Item in Container
Item in Container
• Application which uses the media item library
• Loads multiple container controls to manage multiple media items on the board surface
NC
NC
NC
Shared Dictionary Server

Figure 5.7b – Media Item Architecture with the Media Item Library

When any ActiveX control is installed, the Microsoft Windows operating system tracks it through its system registry, where the control stores some information about itself. The NC board can inspect the registry at any time to discover which media items are available, retrieve their human-readable names, and find where they are located. As with most standard ActiveX controls, these visual components gather input and display visual output. However, media item controls also implement the *IMediaItem* interface (first item in Figure 5.7a, also in Layer 1 in Figure 5.7b), a special software interface that allows these ActiveX controls to be exploited by other layers in the media item architecture.

The next part of the architecture is an Item Container, which is also an ActiveX control except that it also implements *MediaItemHost* capabilities (second item in Figure 5.7a, also represented as layer 2 in Figure 5.7b). This container dynamically loads media items, and it also acts as a kind of proxy between media items and the NC board (Layer 3 in Figure 5.7b). The end result of this design is that media items are never in direct contact with the NC Board client. Instead, the NC Board only needs to know how to display and interoperate Item Containers, while the Item Containers only needs to know how to load and display Media Items while interoperating with both the Media Item and the NC Board. The NC board never actually concerns itself with the contents of media items: the appearance and disappearance of media items on the NC is managed in multiple container controls and signalled by changes to item metadata contained in the shared dictionary.

Before going into further details about the above interfaces, the concept of media item 'ownership' first alluded to in section 4.2.4 must be explained. ActiveX controls are interface components, and they are typically used to display output and gather user input. This proves a natural mechanism for having media item ActiveX controls support groupware input and output in place i.e., without the need for a separate client interface for each media item. The problem is that media items may be 'owned', where its creator should have different input permissions (or even a different view) when compared to those others who just view it. For example, *Sticky Notes* (see section 4.3.1) are owner-specific: only a note's poster (thus, the creator) should be able to change its contents. The creator may also see controls within the Sticky Note not visible to others e.g., a button to change the caption as displayed within the note. Having a notion of 'owner' and 'viewer' introduces a certain level of modality as to who can manipulate input on specific items. To this end, the architecture recognizes three types of media items: 'master', 'slave', and 'un-owned' items. When a media item is loaded on the NC, it will be loaded as one of these three types.

'Master' and 'slave' items are two different modes expected on one type of media item. The 'master' is the media item mode for the item creator, while the 'slave' is the mode for other viewers. That is, when a person posts a media item on their own NC board, it is created in the 'master' mode. What appears on other peoples' NC clients is the same media item in the 'slave' mode. This distinction allows developers to design media items such that the people posting information have exclusive access to modifying it, but others can still view it. For example, when a *Video Snapshot* (see section 4.3.2) is posted, only the person to whom the video belongs should be able to control when the video is running, blurred, or paused. The master instance will show controls for doing these actions, but the slave will not.

Finally, an 'un-owned' item is an item whose contents are equally accessible by anyone. When a person creates an un-owned item, it appears identically on all displays, and anyone can control or modify its contents equally (for example, the desktop snapshot item in section 4.3.5).

This subsection gave greater detail on how the media item library fits into the media item architecture, as well as the master/slave concept for media items. The next subsection covers the *Media Item Host* control for hosting media items developed as ActiveX controls using the development model as provided by the *IMediaItem* interface, which is described following the host.

5.3.2 Hosting media items using the Media Item Host control

The Media Item Host control is a standalone ActiveX control that can host any media item. That is, it can contain any ActiveX control that supports the *IMediaItem* interface (described in the next section). Acting as a generic wrapper for all kinds of media items, this control makes it possible for the NC board to be built using only a single type of control to represent and manage a diverse number of media items. The host control itself provides methods that mirror the IMediaItem interface, but it mainly only delegates the calls to the appropriate implementation on the contained media item.

On the NC board itself, media items appear as small windows, including a small caption and title bar (see section 4.2.4). The actual construction of this arrangement as it appears on the NC board using the media item host control is illustrated in Figure 5.8, below. As described above, each media item is implemented as a separate ActiveX control (Figure 5.8a). These specialized controls are loaded and hosted inside a *media item host* (Figure 5.8b). The item host has no visual interface itself: it only acts as a transparent proxy between media items and the NC board. For this reason, the NC board adds a UI wrapper that makes each host resemble a window (Figure 5.8c), placing the item management tools discussed in chapter 4 directly on each media item. The most important component of this hosting system and the focus of people using it is the media item. To this end, the host and the UI wrapper will both automatically size themselves to whatever size the media item takes, creating the combined interface seen by each NC board user (Figure 5.8d).

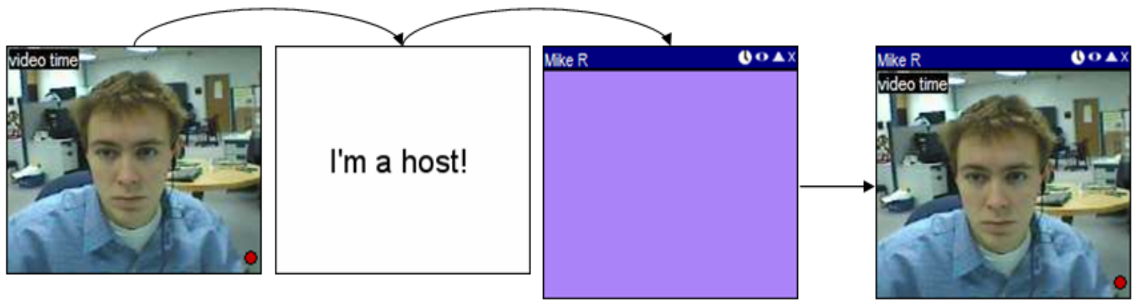


Figure 5.8a - A media item ActiveX control

Figure 5.8b - The *media item host*

Figure 5.8c - The item host's UI wrapper

Figure 5.8d - A media item hosted as it appears on the NC

The hosting system allows the NC board to include new media items on the fly. The arrival of a new media item is signalled in the SD server by the arrival of a new media item information structure (see section 5.2.4.2). At this point, the NC board will dynamically load a new host control. The host control is an empty container with no visible GUI of its own. Using the information structure in the SD, the host attempts to load the media item described by its **ProgID** field.

Chapter 4 briefly touched on the user interface for including previously uninstalled media items (see section 4.2.5). This facility is provided directly by the media item host control. If the item described by the **ProgID** field is not installed on the system, the loading process will fail, and the host displays its only user interface: a ‘**download**’ button and a ‘**ready now**’ button. Pressing the ‘download now’ button will cause the host control to retrieve the installation script for the item using the **DownloadURL** field of the item’s structure stored on the SD server. This invokes a standard download and installation script common to most applications in Windows, and thus familiar to users. Once the installation is complete, pressing the ‘ready now’ button will load the new item (this trigger is needed is because the host control cannot automatically know when installation is complete).

This scheme, coupled with the *IMediaItem interface* – presented in the next section – completes the architecture for dynamically loading media items. Through the host, an application (like the NC board) can load and host any media item, and not concern itself with the contents of the item at all. This allows new items – developed separately from the NC board – to be added quickly and on the fly, without the need to modify the hosting application (the NC board).

5.3.3 The IMediaItem Interface

This section details the *IMediaItem interface* and how a developer would use it to create a media item.

The IMediaItem interface transforms an ActiveX control into a media item. Through *interface inheritance,* the ActiveX control implements and exposes the methods and properties defined by the IMediaItem software interface. These methods and properties are then used by the NC board through the media item host container (see the previous section) to interact with it. Other functionality can then be added: as long as this software interface is supported, the media item contract is fulfilled and the ActiveX control becomes a media item.

I explain the IMediaItem interface from a programmer's perspective. Because most media items have a very similar (or even identical) implementation of the methods and properties that comprise the interface, we provide programmers with a *template media item* as a starting point[3]. We expect programmers to use this template to transform a Visual Basic 6 ActiveX Control project into a testable media item, where they use the template to 'fill in the blanks' as they create their items. This allows them to focus on unique aspects of the media item interaction design rather than low-level common coding.

While the following code may appear complex, the reality is that its only visible in the template because Visual Basic 6.0 is not a true object-oriented programming language. If it were, much of this code would be hidden in parent objects.

5.3.3.1 Initial Media Item Setup

The media template uses or tracks several common objects and variables, all of which are declared at the beginning of the code for the template item. These declarations are shown in Figure 5.9. The first line imports the IMediaItem interface so that it can be implemented by the ActiveX control. The second line declares SD, an instance of the Collabrary used by

```
1  Implements MediaItem.IMediaItem

2  Dim WithEvents sd As Collabrary.SharedDictionary
3  Dim m_Information As Collabrary.Struct
4  Dim EventObject As MediaItem.MediaItemEventObject
5  Dim m_MediaPath As String
6  Dim m_MediaMode As MediaModes

7  Private Const MediaItemTemplateProgid = "Template.NCMediaItem"
8  Private Const MediaItemTemplateMimetype = "text/ascii"
9  Private Const MediaItemTemplateURL = "http://install.msi"
10 Private Const MediaItemTemplatePath = ""
```

Figure 5.9 – Initial media item declarations

[3] The template was created in cooperation with Kathryn Elliot, a research assistant who worked under my supervision for this project.

the media item to post and share information. A Collabrary structure is declared which is used to house information about the media item as described in the NC data model (section 5.2.4.2). Next, an event object is declared. This is used by the media item to fire user interaction events that can be synchronized across the three layers of the architecture (e.g., so the NC board can know when an item is clicked, Figure 5.7b). For lines 4 and 5, the `m_MediaPath` variable refers to the **RootPath** value of the information structure (see section 5.2.4.2), and the `m_MediaMode` stores the user mode in which the item was started (the possible values of which are 'master,' 'slave,' and 'un-owned,' as outlined in section 5.3.1). The four remaining constants are posted at the top of the template so they can be set once, and are used by the programmer to fill in the information structure about the item and to use them within the program.

5.3.3.2 IMediaItem methods for starting and stopping

Starting and stopping a media item comprises the main portion of a media item template. Once the media item is started, very little architectural maintenance is required. The basic starting and stopping pattern is common across all media items, as defined by the architecture design. Consequently, the template supplies a basic 'shell' that creates a fully operable (albeit empty) media item. Developers are expected to add their own item-specific setup code at specified points in these methods to support any item-specific interaction they want to include.

Figure 5.10 shows the implementation for the `IMediaItem_MediaStart` method. The first things it accomplishes are storing the *media path* (`m_MediaPath`, the **RootPath**) and user mode (`m_MediaMode`). These values are supplied through the layers of the architecture from an external calling application like the NC board. It then initializes the event object and media item information structure. Following that, a SD connection supplied to the media item through this method call is saved in a local copy for convenient use. Note that a new connection is not established; instead, media items use an existing connection so it can be integrated into the hosting application with minimal overhead. From here, the execution stream of the method splits depending upon which user mode the media item is started.

If a ‘master’ item is started, then the item information structure is built (section 5.2.4.2). This is done partly with information supplied from a calling application (such as the NC board) through the architecture (in the case of **Owner** and **Rootpath**) and partly with information provided by the media item (for the **ProgID, MIMEType, DownloadURL** and **RelativePath** fields). At this point, the developer is expected to write code for setting up the unique aspects of their media item. This could include things like

```
Public Sub IMediaItem_MediaStart(ByVal MediaMode As _
    MediaItem.MediaModes, ByVal ShDict As Object, _
    ByVal MediaPath As String)

    On Error Goto SetupFailed
    m_MediaPath = MediaPath
    m_MediaMode = MediaMode

    Set m_Information = New Collabrary.Struct
    Set EventObject = New MediaItem.MediaItemEventObject

    If Not ShDict Is Nothing Then
        Set sd = ShDict
        Select Case m_MediaMode
            Case meMaster:
                'post media item information
                m_Information("owner") = sd.Me
                m_Information("rootpath") = m_MediaPath
                m_Information("progid") = MediaItemTemplateProgid
                m_Information("mimetype") = MediaItemTemplateMimetype
                m_Information("downloadurl") = MediaItemTemplateURL
                m_Information("relativepath") = MediaItemTemplatePath

                'TODO: ADD CODE FOR MASTER SET UP HERE

                sd(m_MediaPath) = m_Information
                sd(m_MediaPath & "/.transient") = sd.Me

            Case meSlave:
                'retrieve media item information
                Set m_Information = sd(m_MediaPath)

                'TODO: ADD CODE FOR SLAVE SET UP HERE

            Case meNoOwner:
                'TODO: ADD CODE FOR UNOWNED MEDIA ITEMS HERE

        End Select
    Else
SetupFailed:
        IMediaItem_MediaStop
    End If
End Sub
```

Figure 5.10 – The MediaStart method

setting up SD subscriptions, or starting up desktop web cameras and timers. It is only after this that the item structure is posted to the SD server. This ordering is important because it is the appearance of the new structure in the SD server that signals to the NC board that a new item has been added. By posting the structure following the setup code, a media item developer will know that any required setup (both locally and server-side) is completed before anyone else sees the item.

If a ‘slave’ item is started, the information structure signalling the arrival of the new item is retrieved (this will have been posted by the master instance as described in the previous paragraph). This allows the slave to have access to the same setup information as the master. Most importantly, the owner information is available, enabling a later connection to be created between someone viewing the ‘slave’ item and the person who posted the ‘master’ item. After retrieving the structure, the programmer can add any setup, most of which will likely mirror or complement the setup code in the ‘master’ section.

Finally, unowned items only need setup code. The information structure is not retrieved because the **owner** field in the structure is not required.

The `IMediaItem_MediaStop` method (illustrated in Figure 5.11) is much simpler. First, the media item developer cleans up any of their item-specific objects that must be removed before their media item is unloaded. This include things like stopping timers, unloading any auxillary dialogs that were used by the media item, and removing SD subscriptions. Next, if the media item was a ‘master’ item, it will esentially clean up after itself, removing

```
Public Sub IMediaItem_MediaStop()
    On Error GoTo DoneAnyways

    'TODO: ADD CODE TO CLEAN UP ITEM-SPECIFIC RESOURCES

    If m_MediaMode = meMaster Then
        sd(m_MediaPath) = Nothing
        sd(m_MediaPath & "+") = Nothing
    End If

DoneAnyways:
End Sub
```

Figure 5.11 – The MediaStop method

the item information structure (thus signalling to other NC board clients that the item is being removed) and any keys that were placed by the item. The ‘slave’ and ‘unowned’ items do not have to do anything.

5.3.3.3 Objects Returned Through IMediaItem and Related Methods

The IMediaItem interface implementation returns three objects that can be used by other layers in the architecture. Figure 5.12 shows the software implementation of these, plus one method that is directly related to one of the objects.

The first object is acquired via the `IMediaItem_MediaEventsObject` property. This object allows user interaction events (such as mouse clicks and keyboard actions) to be synchronized through the different layers of the architecture. Figure 5.12 shows that the `EventObject` variable is first initialized (if this has not already been done), and then the object itself is returned. Because the implementation of this method will be the same for every media item, the developer need never modify this definition.

The next object is acquired through the `IMediaItem_MediaMenu` property. The idea here is that the media item developer can build a context menu using the ShorcutMenu object (a component in the media item library) that can be passed through other layers in the architecture. The object can then be used on its own to construct a menu, or appended

```
Public Property Get IMediaItem_MediaEventsObject() As Object
    If EventObject Is Nothing Then _
        Set EventObject = New MediaItem.MediaItemEventObject
    Set IMediaItem_MediaEventsObject = EventObject
End Property

Public Property Get IMediaItem_MediaMenu() As Object
    Dim scm As New MediaItem.ShortcutMenu
    'TODO: build your context menu here
    Set IMediaItem_MediaMenu = scm.MenuHeirarchy
End Property

Public Sub IMediaItem_MediaHandleMenu(ByVal Selection As String)
    ‘TODO: insert code to handle context menu callbacks if necessary
End Sub

Public Property Get IMediaItem_MediaItemInfo() As Object
    Set IMediaItem_MediaItemInfo = m_Information
End Property
```

Figure 5.12 – IMediaItem objects and related methods

as a sub-menu within another context menu hierarchy (as in the case of the NC Board). Figure 5.12 shows that an instance of the ShortcutMenu object is declared (the `scm` variable), following which the media item developer would define their item-specific menu hierarchy. The method immediately following the property is the `IMediaItem_MediaHandleMenu` method. This method directly maps to the `IMediaItem_MediaMenu` property: it should be written to handle the menu cases defined in the hierarchy. As before, these implementations will work as is, and there is no requirement that a menu hierarchy be further defined.

The final object is acquired through the `IMediaItem_MediaItemInfo` property. This object contains the information structure described in section 5.2.4.2. It is provided for convenience to other layers of the architecture. Instead of other layers having to figure out where the information structure for a particular item is stored in the SD server, the entire information structure can be accessed directly as a property of the item.

5.3.3.4 Other IMediaItem Properties and Methods

The remaining parts of the IMediaItem interface are two simple properties and one method, illustrated in Figure 5.13. The first – `ImediaItem_MediaDownloadURL` – lets a developer of an application (such as the NC board) re-assign a download URL to the media item. This is useful for a local network distribution as it allows an item installation script to be

```
Public Property Let IMediaItem_MediaDownloadURL(ByVal RHS As String)
    m_Information("downloadurl") = RHS
    If Not (sd Is Nothing) Then
        If sd.Status = Opened Then
            sd(m_MediaPath) = m_Information
        End If
    End If
End Property

Public Property Get IMediaItem_MediaItemMode() As MediaItem.MediaModes
    IMediaItem_MediaItemMode = m_MediaMode
End Property

Private Sub IMediaItem_MediaResize(ByVal Width As Long, _
    ByVal Height As Long)
    'TODO: insert resizing code for media item here
    EventObject.RaiseSizeChanged
End Sub
```

Figure 5.13 – Other methods and properties of IMediaItem in the Template

moved for retrieval to a local network path, as opposed to having clients retrieve the scripts from the default scattered URLs potentially all over the world wide web. This property's implementation need never be altered.

The second property, which also need never be altered, is the `IMediaItem_MediaMode` property. Its sole function is to return the user mode in which the media item was started (master, slave, unowned). This is used by other layers of the architecture to quickly identify item ownership.

Finally, the `IMediaItem_MediaResize` method is implemented by the media item developer to handle media item resizing. If a user attempts to resize a media item (i.e., from the NC board interface as described in chapter 4, section 4.2.4), the set of new size dimensions are gathered and this method is invoked. The developer inserts code to resize the item's visual components. By calling the `EventObject.RaiseSizeChanged` method, the media item signals to the other layers (most importantly layer 2, the item container in Figure 5.7b) that its size has changed and appropriate actions should be taken (for example, resizing the item container). Leaving the method as is means that resize requests will never be addressed and that the media item will always remain the same size.

5.3.3.5 IMediaItem Interface Summary

In this section, I described the design for layer 1 (Figure 5.7b) in the architecture, where a media item is defined as an ActiveX control that implements the IMediaItem software interface. As ActiveX controls, the programmer can create media items that gather input and display output. The interface was described from the programmer's perspective, where I walked through a template supplied to developers to help them create media items. Unfortunately, because Visual Basic 6.0 is not a true object-oriented language, more code is visible than necessary; some of this could have been hidden with inheritance.

While apparently complex, the template is easily learnable and usable by average programmers. The next section details the results of distributing this development model to an undergraduate computer science class, validating its double duty design not only as an architecture, but as a toolkit for exploring novel groupware interfaces in media items.

5.4 Validating Ease of Development

The fourth goal stated at the beginning of this chapter called for the development model, presented as the media item template in section 5.3.3, to be simple enough for average programmers to quickly and easily prototype novel designs for media items.

To this end, the media item template was distributed to a fourth year undergraduate interaction design course in computer science. They were instructed on the NC, the shared dictionary, and how to use this template through a single lecture. Students had no prior knowledge of the shared dictionary, the NC, or even programming with the model-view-controller paradigm using a distributed data / notification server. Many students were relative newcomers to the VB6 language used to write the template. Most had never written groupware before. Given these conditions, we label these students as 'average programmers' as they had no prior knowledge or experiences developing NC media items or groupware.

After two weeks, the fifteen students in the class each successfully produced and demonstrated a novel item design, some which were far more sophisticated than was expected. This section illustrates five of the fifteen designs the students produced after being given a brief introduction to the system and having two weeks of development time[4]. The important point for validation is that no changes to the new NC board client (discussed in chapter 4) were necessary for the students to carry out their designs.

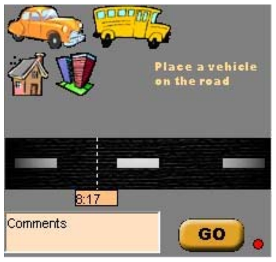
[4] Of course, this was not two weeks of full time work on this design. Most students took 5 courses, each with other assignments that had to be fulfilled. However, we did not track the hours spent by each student on the media item assignment.

5.4.1 Transportation Pool Item

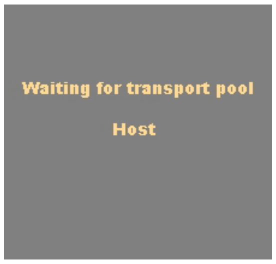
Alan Flanders designed a media item to organize and track times for transportation pools that would help groups of people manage carpools and/or get together on transit systems. Figure 5.14 shows how the media item operates for a group of people using the NC, where the 'master' view is shown on the left (as seen by the creator) and the 'slave' view on the right (as seen by all others).

The owner chooses a transportation medium and a destination by dragging them from the palette (step 1, top left) to the road. They do the same with a destination, dropping the destination on a time later on in the day. After optionally adding a comment, they click the 'Go' button (bottom right of the item) and the route is broadcast to the other users. Other people can join the route by pressing the 'Join' button, and they will see their name on the list. When the departure time approaches, a departure countdown is shown and an alarm raised when the departure time arrives.

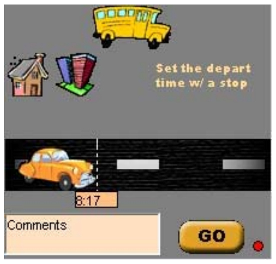
Master **Slave**



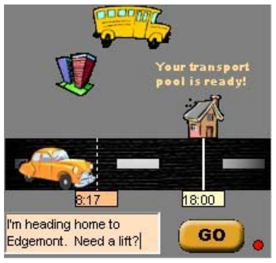
1. Master is posted. The current time is shown on the road.



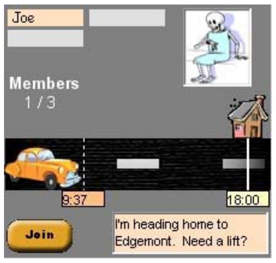
2. Slave appears in response, waits for the master to post a route.



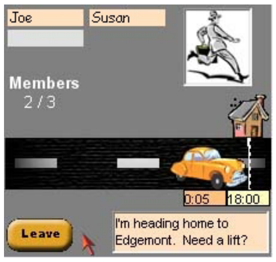
3. Master drags and drops a transportation medium on the road (car or bus).


Your transport pool is ready!
8:17
18:00
I'm heading home to Edgemont. Need a lift?
GO

4. Master drags and drops destination (home or work) at a time, adds a comment. Then click **Go** button.



5. The proposed route appears to other users. Time changes to countdown.

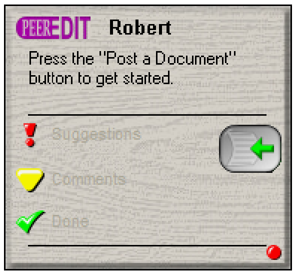


6. Other people press **Join**. Their names appear. Throughout day, departure countdown continues.

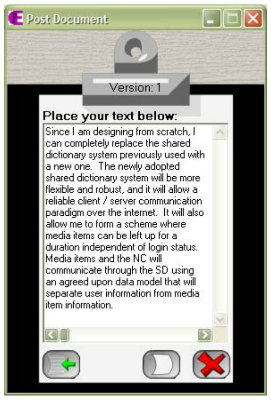
Figure 5.14 - The Transportation Pool Item

Step 1

Master **Slave**

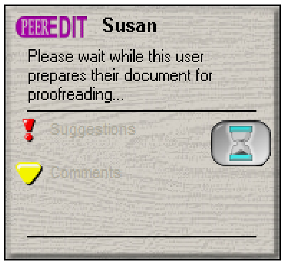

PEEREDIT Robert

Press the "Post a Document" button to get started.



Place your text below:

Since I am designing from scratch, I can completely replace the shared dictionary system previously used with a new one. The newly adopted shared dictionary system will be more flexible and robust, and it will allow a reliable client / server communication paradigm over the internet. It will also allow me to form a scheme where media items can be left up for a duration independent of login status. Media items and the NC will communicate through the SD using an agreed upon data model that will separate user information from media item information.

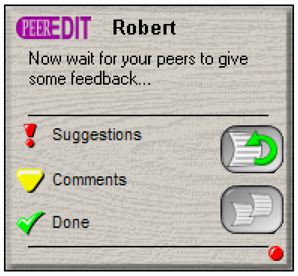

PEEREDIT Susan

Please wait while this user prepares their document for proofreading...

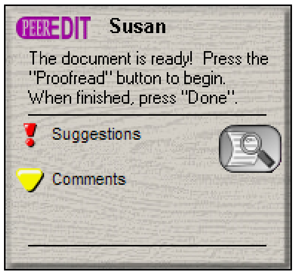
When the Peer Edit item is posted, the owner posts a document by pressing the post document button (right side of the Master item) and using the document window (center). The slave item shows that it is waiting for a document to be posted.

Step 2

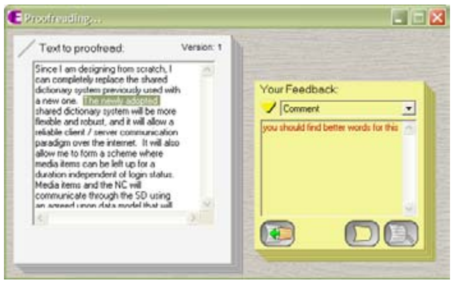
Master **Slave**


PEEREDIT Robert

Now wait for your peers to give some feedback...


PEEREDIT Susan

The document is ready! Press the "Proofread" button to begin. When finished, press "Done".



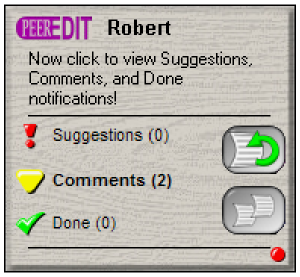
Since I am designing from scratch, I can completely replace the shared dictionary system previously used with a new one. The newly adopted shared dictionary system will be more flexible and robust, and it will allow a reliable client / server communication paradigm over the internet. It will also allow me to form a scheme where media items can be left up for a duration independent of login status. Media items and the NC will communicate through the SD using an agreed upon data model that will

you should find better words for this

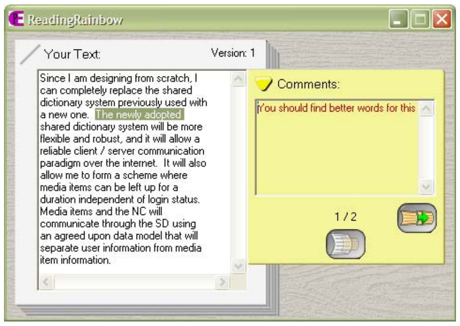
After the master posts a document, it shows the owner no comments or suggestions have been posted. A slave can post a comment or suggestion by clicking either caption on the item (right side of the slave item) and using the revision window (right)

Step 3

Master **Slave**

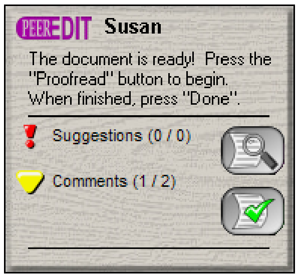

PEEREDIT Robert

Now click to view Suggestions, Comments, and Done notifications!



Since I am designing from scratch, I can completely replace the shared dictionary system previously used with a new one. The newly adopted shared dictionary system will be more flexible and robust, and it will allow a reliable client / server communication paradigm over the internet. It will also allow me to form a scheme where media items can be left up for a duration independent of login status. Media items and the NC will communicate through the SD using an agreed upon data model that will separate user information from media item information.

You should find better words for this


PEEREDIT Susan

The document is ready! Press the "Proofread" button to begin. When finished, press "Done".

When someone posts a comment or a suggestion, the master control is notified and plays an audio cue (master media item, in middle). Clicking on the comments or suggestions caption will raise the review dialog (center window) where the highlighted text and comment is displayed.

Figure 5.15 - the Peer Edit media item

5.4.2 The Peer Edit Item

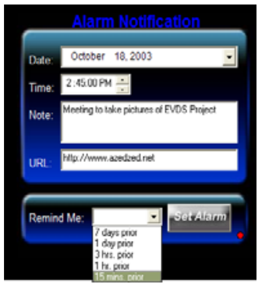
Roberto Diaz-Marino designed the Peer Edit item that provides an online versioning system for text documents (Figure 5.15 previous page). Through this media item, people post a text document, and people viewing the item can respond to the document with comments and suggestions.

When first posted, the owner adds a text document to the master using the clipboard dialog (Step 1). Until this point, slave items display that they are still waiting for a document. When the document is posted, the slave interface adds controls so others can add comments and suggestions via a 'review' window (Step 2). Similarly, the master interface now allows the owner to view these comments and suggestions. Clicking on the comment or suggestion caption allows the owner instant access to the comments in the review window (Step 3).

5.4.3 The NC Alarm Item

Christina Escabillas constructed the NC Alarm item that allow a person to post an event pertinent to everyone who may be viewing the NC, and to trigger an alarm before the event comes due (Figure 5.16, below).

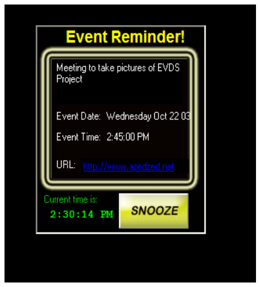
When the owner posts the item, they are presented with an event interface where they



October 18, 2003
2:45:00 PM
Meeting to take pictures of EVDS Project
http://www.azedzed.net
7 days prior
1 day prior
3 hrs. prior
1 hr. prior
15 mins. prior

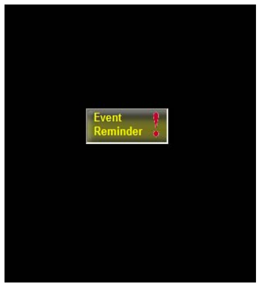
Step 1: The owner sees a scheduling window where they **set an alarm** for an appointment.

Step 2: Everyone sees a blank screen while the alarm is counting down.



Meeting to take pictures of EVDS Project
Event Date: Wednesday Oct 22 03
Event Time: 2:45:00 PM
URL: http://www.azedzed.net
Current time is:
2:30:14 PM

Step 3: The event reminder shows with an audio cue to remind people connected to NC.



Step 4: Pressing the **Snooze** button dismisses the reminder temporarily.

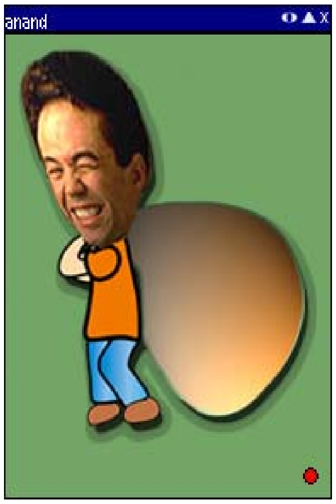
Figure 5.16 - The NC Alarm item

can set up an alarm for an event happening sometime in the future (Step 1). The date dropdown (middle bottom, step 1) allows them to pick a date directly from a calendar view. Until this point, everyone else sees a blank item (Step 2). When the appointment time approaches, the appointment particulars appear on everyone's display, and a light audio cue is played as an alarm (Step 3). Pressing the 'Snooze' button from this state (Step 3, bottom right) allows people to individually defer the alarm and be reminded a short time later (Step 4). This very simple and common idea in scheduling and personal management software is made more interesting by it being situated in a public space dedicated to informal awareness and casual interaction.

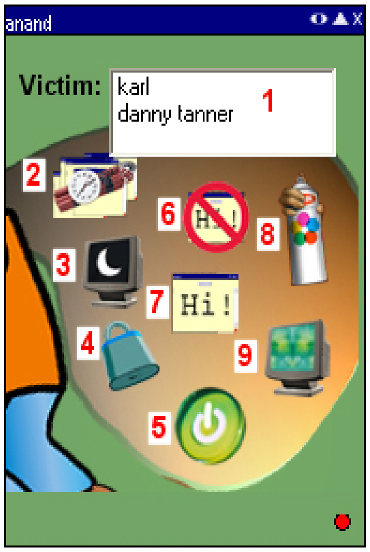
5.4.4 The NCHack Item

Anand Agarawala took a different creative direction with his NCHack media item (Figure 5.17, below). NCHack takes advantage of the SD and other media items on the NC board by modifying the contents of the SD to change the appearance of items owned by other people. When the item is first posted, Gilbert and his bag of tricks appears (Figure 5.17a). The person posting the item is the only one who sees it. When the item owner clicks on the bag, they are presented with a list of users to choose as victims, labelled as 1 in Figure 5.17b. To use NCHack, the owner selects a person who is logged in from the list as a target and then uses one of the tools on the palette, labelled as follows in Figure 5.17b:

2. Random spoof note – this tool will post a random message (a set of which is built into the NCHack item) in a *Sticky Note* as if it were the person selected as victim.
3. Load screensaver – selecting this tool will bring up the screensaver on the workstation of the person selected as victim.
4. Lock workstation – this will lock the workstation of the person selected as victim.
5. Power off workstation – this tool will shut down Windows and power off the workstation of the person selected as victim.

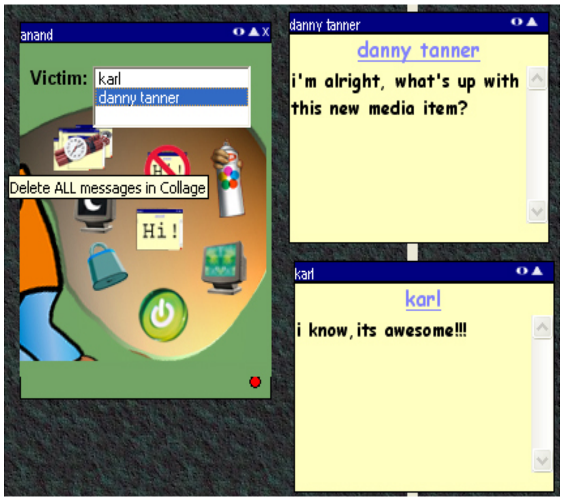


A: The NCHack item posted, Gilbert and his bag of tricks.



B: Clicking on Gilbert's bag yields a toolkit of toys:

1. List of connected users to target
2. Random spoof note
3. Load screensaver
4. Lock workstation
5. Power off workstation
6. Kill all spoof notes
7. Custom spoof note
8. Video vandalize
9. Change desktop wallpaper



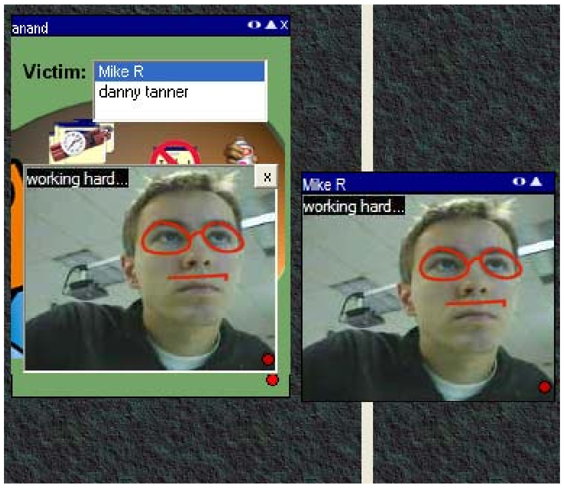
danny tanner

i'm alright, what's up with this new media item?

karl

i know, its awesome!!!

C: Using the NCHack item to post spoof notes from other people connected to NC.



D: Using the NCHack item to take someone's video snapshot and apply graffiti.

Figure 5.17 - The NCHack Item

6. Kill all spoof notes – this allows the owner of the NCHack item to remove all of the spoof notes they have posted.

7. Custom spoof note – this posts a *Sticky Note* seemingly from the victim to the NC board. The owner of the NCHack item can then type messages in the *Sticky Note* as if it were their own (illustrated in Figure 5.17c).

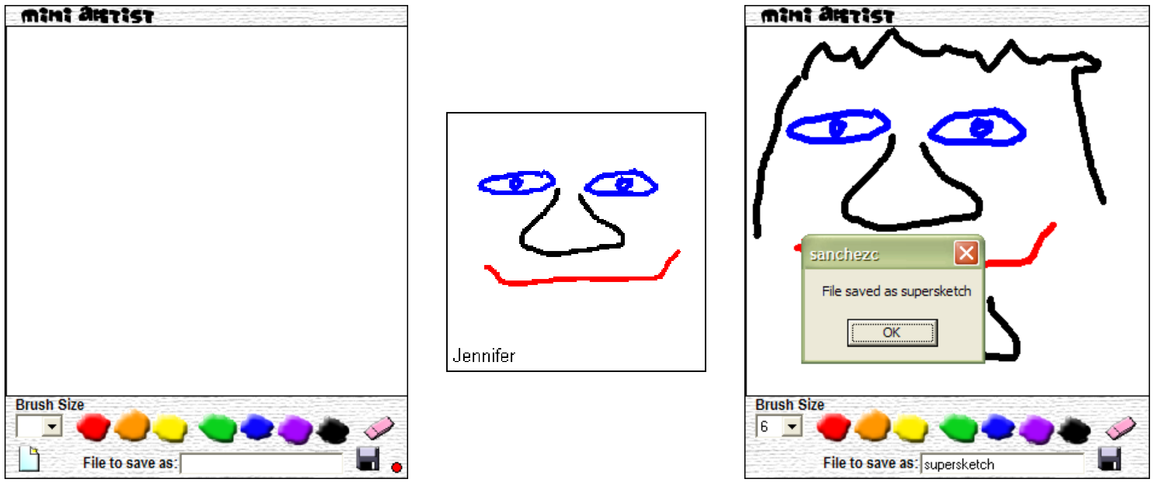
8. Video vandalize – if a victim has a *Video Snapshot* posted, this tool allows the owner of the NCHack item to mark the target's video image with graffiti (illustrated in Figure 5.17d).

9. Change desktop wallpaper – this will change the desktop wallpaper of the victim to be a picture of the NCHack item.

These tools were all built for fun and are not intended for malicious use. It should also be noted that the tools marked 3, 4, 5 and 9 all take advantage of system calls to the Win32 API, the system library in Windows, and they also require that the target of the trick have the NCHack item installed on their system. While done for fun, it does suggest that a commercial version of the NC should have security safeguards built into it.

5.4.5 The Mini Artist Item

For our final example, Rosemary Sanchez built the Mini Artist media item (Figure 5.18) which inspired the *SketchPic* item described in chapter 4 and which appears as a stock media item in the NC distribution. The Mini Artist is a simple groupware sketch pad that lets people connected to NC collaborate on a drawing and save the results in an image file.



File saved as supersketch

File to save as: supersketch

A: Blank master item posted with a blank canvas and tool palette

B: Slave item shows thumbnail and name of posting artist

C: Type a file-name and click the disk icon to save a copy of the drawing

Figure 5.18 - The Mini Artist media item

Figure 5.18a shows the item as it first appears to the owner when they post it. The tool palette at the bottom of the canvas allows the artist to change colours and brush sizes, choose an erase tool, clear the canvas (only available to the owner of the item) and save the drawing to an image file. Other people first see only a thumbnail of the drawing session with the name of the owner on it. This thumbnail is actively updated as the drawing evolves (Figure 5.18b). Double clicking the thumbnail allows others to join in on the sketching session. After drawing for a while with several people, owners and artists who have joined the drawing session can all save the drawing as an image file using the text box and disk icon at the bottom of the canvas. When the image is saved, the item displays a message box with the name of the file (Figure 5.18c). This item is a very simple and elegant example of embedding groupware sketch tool (e.g., Greenberg & Bohnet, 1991) directly into the NC as a media item.

5.5 Summary

In this chapter, I have presented my new design for the Notification Collage architecture (section 5.1.2), motivated from four redesign goals (section 5.1.1):

1. **Use a robust shared dictionary that lets the Notification Collage distribute, store and control both simple data structures and multimedia across the internet.** I have adopted the *Collabrary shared dictionary* (section 5.2), a heavily featured shared dictionary system. This robust client/server system works over the internet and provides important control of simple and multimedia data such that media item lifetime can be controlled independent of login status. I have also designed a data model that is used by the overall media item architecture to capitalize on the new shared dictionary system.

2. **Media items should be dynamically added to the Notification Collage board without requiring recompilation or re-installation.** I have presented the *IMediaItem interface*, which when implemented within an ActiveX control creates a media item. I have done this by describing the media item template (section 5.3.3),

which is used as the development model for building new items. As ActiveX controls, media items exist as separate and distinct objects from the NC board. They are loaded dynamically into the NC board by using the *media item host control* (section 5.3.2), an ActiveX control which can contain media items. This host provides the layer of abstraction necessary to completely separate the development and deployment of media items from that of the NC board.

3. **A media item should serve as both input and output.** This goal was important to remove the excess of client programs that were needed to run the initial NC (presented in chapter 3). As *ActiveX controls*, media items are now able to incorporate both end-user input and output.

4. **Media items should provide a simple programming model so that average programmers can rapidly prototype new media items.** The development model was validated by giving undergraduate students in an interaction design course the template and only modest instruction. We saw them create and implement fifteen different media item designs, some which were quite sophisticated. Five examples of these were presented in section 5.4. However, I believe that an even simpler programming model could be developed if the NC were re-implemented in a true object-oriented language.

Chapter 6. Conclusion

This thesis concerned the design of a tool that supports informal awareness and casual interaction in a community of intimate collaborators. In particular, I described the evolving development of the user interface, the architecture, and the implementation of the *Notification Collage* (NC). I have shown how the initial version of the NC proved – in spite of its shortcomings – a successful medium for a partially co-located and partially distributed community of people to stay in touch (chapter 3). A redesigned NC addressed these shortcomings. The user experience was illustrated in chapter 4, where its most significant difference from the previous version was that the board served as both input and output. Chapter 4 also included an explanation of how I designed a stock set of media items – each giving community members different awareness information and opportunities for moving into casual interaction – to be included with the distribution of the NC board. The redesign was accompanied by a new Media Item architecture (chapter 5) that also supported how programmers could build new media items and add them onto the NC on the fly. The ease of programming with this architecture was tested by having an average group of programmers use it to quickly create and deploy novel media item designs (chapter 5).

In this final chapter, I comment on the success of the NC as a groupware system and architecture, and how several projects have spun off from this research. I then conclude with future directions for this research, and by revisiting my original research problems and goals to highlight my contribution.

6.1 The Successes of the NC board

6.1.1 Usage

As of Winter 2004, several generations of the NC have been in active use in our community for approximately three years. In that time, membership in the NC community has changed considerably, with new members arriving and other members leaving. One of the successes of the NC in this regard has been its acceptance. New members in the community have quickly adapted it into their work practises and they have continued to use it consistently. As first mentioned in chapter 3, when people saw others interacting through the NC board, they immediately wanted to be able to participate as well. This has lead to widespread – albeit not universal – use of the system by our research group and beyond.

The NC board successfully bridged distance to allow remote community members to participate in our research group. In chapter 3 I mentioned that one of the initial reasons for developing this system was that it allowed telecommuters to stay in touch with those residing in the physical laboratory more effectively via the internet. This proved true in practice. Many group members occasionally working from home as telecommuters, regularly connect to the NC. Over the last three years, some community members left the laboratory for distant locations due to work internships, travel opportunities and sabbaticals in other cities and countries. To overcome the distance barrier, they connected to the NC board from their new locations. This let them maintain a collaborative link with their usual workmates. In addition, people from outside our physical community joined the group via the NC. For example, one current community member became friendly with group members after meeting at a conference in another city. Since then, she 'hangs around' our laboratory via the NC, connecting to it almost daily in spite of residing in a different country and being almost 3500 kilometres away. During this time, visitors to our physical laboratory, particularly research interns from Germany, have continued to connect and collaborate with laboratory members via the NC. Some do this from their homes, from their workplace, and even during travels.

The strongest testament to the success of the system is that the NC board has become the most actively (and continuously) used and observed groupware system in our community, with daily participation ranging from five to fifteen people at any one time.

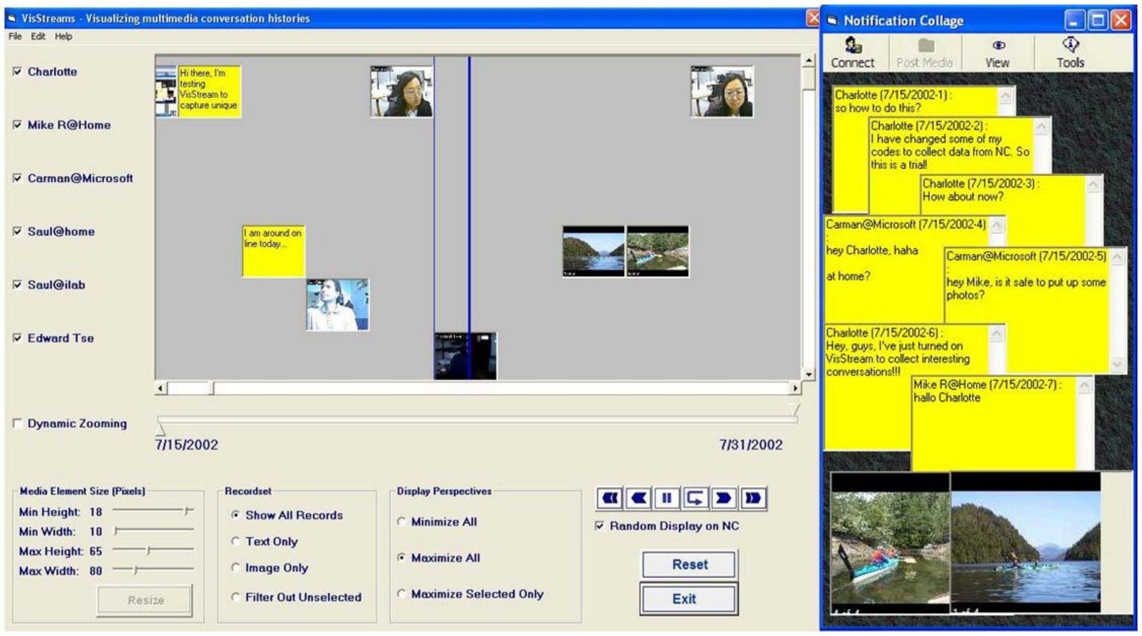
6.1.2 Technical design

From a technical standpoint, the NC has also proven quite successful. First, it has served as a good proving ground for the Collabrary toolkit's *Shared Dictionary* client / server system. It helped define the need for several new features within the Collabrary, and encouraged an API that made it easier for people to capitalize on its shared dictionary when programming multimedia groupware. Media items also exploited the Collabrary's multimedia facilities, validating the Collabrary as an excellent general purpose library. Second, the *media item architecture* turned the NC board into an easily extendable groupware platform. This has made it possible for researchers to rapidly iterate and deploy groupware designs within a collaborative space that already contained a critical mass of users, which in turn meant they could get rapid user feedback of their designs. Third and perhaps most importantly, the addition of novel media items requires little overhead of the people using the NC, enabling them to immediately include new items by clicking a button.

6.1.3 Influences on other researchers

The NC board has also spawned several other research projects and influenced other researchers.

Within our laboratory, Charlotte Tang based her successful Masters of Science thesis on the NC board by building a logging facility called VisStreams (Tang, 2003; Figure 6.1). She did this by taking advantage of the data model in the shared dictionary described in chapter 5. The idea was that the NC board had no way of reviewing past interactions, and that people could potentially benefit from reviewing particular interactions at a later date. In parallel, VisStreams lets sociologists collect usage data of the NC community over time, where they can analyze patterns of use and social relationships within the community. Tang



☑ Charlotte
☑ Mike R@Home
☑ Carman@Microsoft
☑ Saul@home
☑ Saul@ilab
☑ Edward Tse

Hi there, I'm testing VisStream to capture unique

I am around on line today...

7/15/2002

7/31/2002

Media Element Size (Pixels)
Min Height: 18
Min Width: 10
Max Height: 65
Max Width: 80

Recordset
[✓ Show All Records]
Text Only
Image Only
Filter Out Unselected

Display Perspectives
Minimize All
[✓ Maximize All]
Maximize Selected Only

☑ Random Display on NC

Charlotte (7/15/2002-1) :
so how to do this?

Charlotte (7/15/2002-2) :
I have changed some of my codes to collect data from NC. So this is a trial!

Charlotte (7/15/2002-3) :
How about now?

Carman@Microsoft (7/15/2002-4) :
hey Charlotte, haha

at home?

Carman@Microsoft (7/15/2002-5) :
hey Mike, is it safe to put up some photos?

Charlotte (7/15/2002-6) :
Hey, guys, I've just turned on VisStream to collect interesting conversations!!!

Mike R@Home (7/15/2002-7) :
hallo Charlotte

Figure 6.1 – VisStreams and its conversation visualization window (From Tang 2003)

generalized her experiences, where she presented a taxonomy of tasks for the development of logging tools in multimedia groupware.

The NC has also become the cornerstone of several other grants. One (currently in progress) has as a goal to transform the NC into a form suitable for technology transfer, as well as deployment for use by other communities. Another major grant, recently awarded, lists several projects that exploit the NC as is, and as it could be in future versions.

The NC also influenced researchers outside our laboratory, mostly because it was one of the first systems to consider both co-located and distributed communities, and the public display of awareness information on a large display. In particular,

a) Messydesk and Messyboard are applications intended to improve peoples' memories of information as they work (Fass, Forlizzi & Pausch, 2002). They do this by encouraging people to create a context for information using freeform decoration and organization (e.g., remembering that a set of papers about a particular topic was deliberately placed on a particular shelf). Messydesk is an alternate desktop for a Windows system that allows the user to place information fragments (pictures, text and other objects) on their desktop. Wishing to form a group context, Messyboard is essentially a networked version of Messydesk. Similar to the NC, it is positioned both on peoples' personal displays as well as projected on a large display. Unlike NC, Messyboard has the same view for all people connected in order to encourage a group context or storage.

b) The Everyday Computing Lab at the Georgia Institute of Technology has taken inspiration from the NC in their work on semi-public displays (Huang & Mynatt, 2003). They have chosen to support intimate collaborators as the NC does. However, their focus is on co-located groups. This is the reason for the semi-public description of their display: it is public in that the group can view its contents, but private in that only the group can view it. For the system, the designers took active stock of the types of information that people collected on a daily basis, and enhanced it while presenting it on a public display. They include presence information on a group portrait, textual reminders of upcoming events, planned attendance by group members of upcoming events and a collaboration space similar to a whiteboard

c) Microsoft Research has cited NC as an example of a peripheral awareness tool influencing their design of SideShow, first prototyped by Michael Boyle, a graduate student in our lab (Cadiz, Venolia, Jancke & Gupta, 2002). It uses an extensible set of 'tickets' to display awareness information (similar to the media item concept) on a vertical bar on the right side of the screen. Like media items, they are discrete elements of information which can be added or removed on the fly. Each ticket is designed to provide a brief overview of some information type, but can be moused over to reveal a tool-tip providing more detailed information. Unlike NC, SideShow is focussed at individual awareness of information more personal to the viewer as opposed to informal awareness of the community. For example, email inbox or frequent collaborators' online status or local traffic on their route home are all acceptable data sets. SideShow designers recognize that people may also want to see more information than cannot be represented by their default set of tickets, and so like NC also have a Software Development Kit (SDK) for building new tickets.

d) The Plasma Poster network is a series of virtual bulletin boards placed in public spaces (Churchill, Nelson, Denoue & Girgensohn, 2003). Each board is a plasma screen with an interactive layer placed on top of it, making it touch sensitive. To it, information is posted in the form of pictures or posters. The information is sourced from many locations including postings from authenticated users through a web or email interface (e.g., pictures of an event), or sampling from various web pages (e.g., .notices newly published tech reports). Information on the board will change over time, with a different poster appearing every sixty seconds. As well, each item has a maximum lifetime of two weeks, after which they are automatically removed from the system. The posters are meant to be simply viewed, but web links can be followed directly and items of interest can be forwarded to people via email. Unlike NC, interaction over information appearing on the displays occurs outside the system, in face to face interactions happening in common areas inhabited by the posters. The developers of the system have gone even further to increase the similarity of the plasma posters to a real-world bulletin board by adding environmental effects to the display. Items are attached to the board with virtual 'pins' and can be rotated to different orientations or blown about by the wind, just as in real life (Denoue, Nelson & Churchill, 2003).

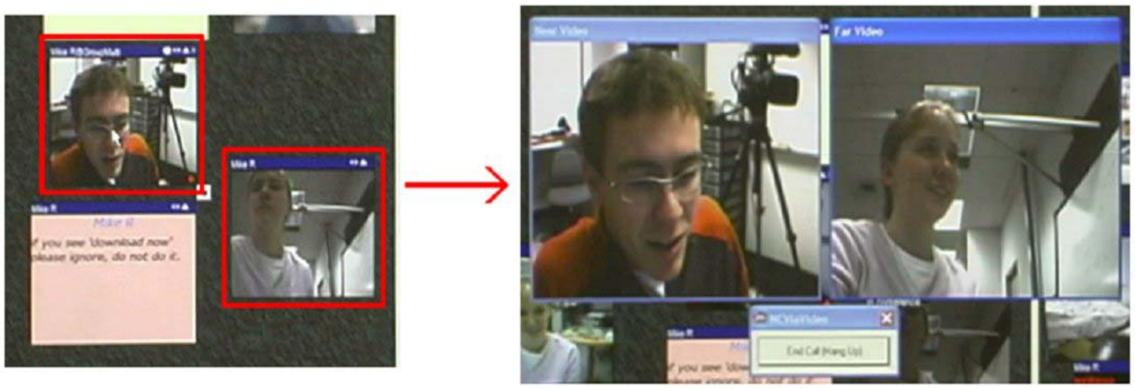
e) The NC has also enjoyed citing by other projects in the research community not mentioned here, solidifying its role as an important contribution in a growing research area.

6.2 Future Work

While the NC board has been tremendously successful, we also recognize the need for further development and research to improve it. A few topics and initial directions are listed below.

6.2.1 Moving from awareness to collaboration and work

We have long recognized that the Notification Collage and its media items should gracefully bridge from awareness to conversation to real-time collaboration and actual work. A hint of how this can be done was already provided by the SketchPic media item



Mike R
If you see 'download now' please ignore, do not do it.

Figure 6.2 – h.323 video connection built into Video Snapshots

(see chapter 4), where people can move from the awareness of a sketching session into a fully developed groupware sketching and chat system directly through the NC board.

This idea is now being refined within the PACE grant project, hosted by the Netera Alliance. I serve as a consultant to this project, and have done some of the work on it. However, others are now taking over – thus its work is described here rather than in previous sections.

First, the video snapshot item (see chapter 4) has been modified so that two people can use it to move into a high fidelity audio/video conversation (Figure 6.2). The redesigned item uses an h.323 compliant camera, which provides hardware based audio and video compression. The software drivers for the cameras also provide their own network links. If one notices that another is available for conversation (perhaps verifying it through a sticky note), one just double clicks that person's video snapshot. Separate larger windows are raised displaying both partners in the full audio/video connection (Figure 6.2 right side). Other people not in the conversation see a label on the snapshot video saying that these two are in conversation.

Second, we would expect that clicking the Desktop Snapshot item should lead to interaction, for example, by allowing both to view and collaborate over a shared live desktop. Kathryn Elliot has prototyped this by linking this media item to third party software that creates a desktop sharing session. SMART Technology's Bridgit software was plugged into a desktop snapshot to add its functionality (http://www.smarttech.com/).

People double-click a desktop snapshot (Figure 6.3 left) to enter into a fully developed desktop sharing facility displayed in a different window (Figure 6.3 right).

For both these projects, the NC itself was not modified. Rather, the media item architecture was exploited to explore these novel item designs by capitalizing on external SDKs built by third party developers.

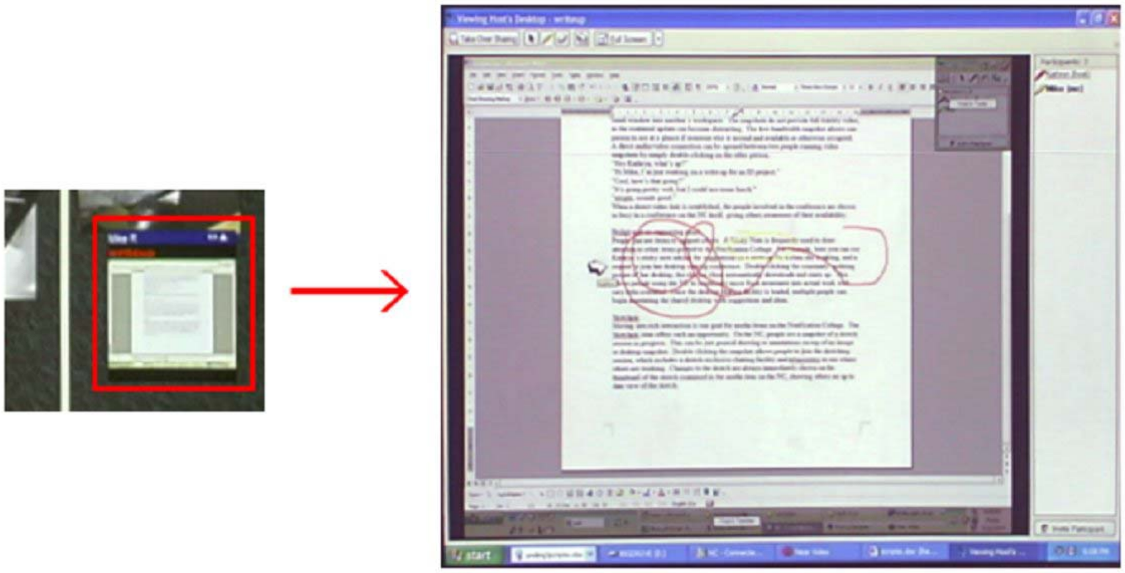


Figure 6.3 - Moving from a desktop snapshot directly to a desktop sharing facility

6.2.2 New metaphors for displaying media items

In retrospect, the collage metaphor is perhaps not optimal for displaying media items. NC users generally tend to prefer an uncluttered space vs. one with many overlapping (and perhaps stale) items. Consequently, they spend time rearranging the display by moving items around rather than letting the collage form. This prompts the question of what display mechanisms and metaphors would better serve the group.

Gregor McEwan, an MSc student in our laboratory, is looking at an alternate display metaphor for his Masters of Science project. He is creating the SideNC (Figure 6.4). Instead of taking up a large portion of a person's display, the SideNC appears as a vertical task bar on the right side of the screen (1 in Figure 6.4). Media items are also designed with several granularities of awareness and interaction built in. For example, the video item can display either the name of the person to whom the video belongs (3 in Figure 6.4) or a very small snapshot frame (2 in Figure 6.4). Moving the mouse over the snapshot makes a larger snapshot video slide out from the side of the SideNC's bar (4 on Figure 6.4), and clicking on that will produce a window containing a full fidelity video link (5 on Figure 6.4). This added granularity allows people to gracefully move from simple awareness to full-on conversation and collaboration.

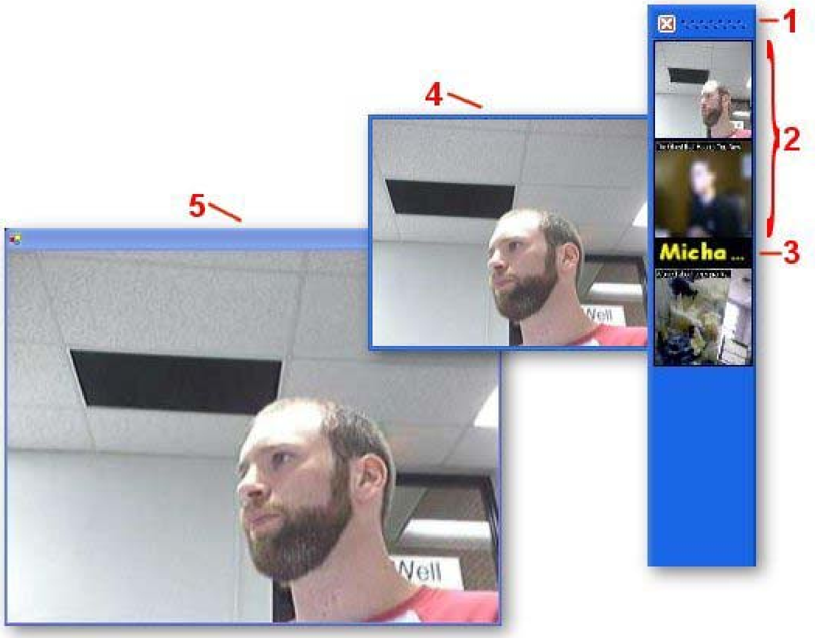


Figure 6.4 – SideNC, with permission from Gregor McEwan

1. The SideNC bar
2. Video Item Snapshots
3. Video Item shrunk to text only
4. Larger Snapshot View
5. Full Fidelity Video Link

6.3 Research Problems and Goals Revisited

In chapter 1, I listed three research problems and goals that I would address in this thesis. In this section I revisit each one, commenting on what I have done to address them.

4. The first problem concerned ***impoverished awareness of the community.*** I stated that present-day tools for informal awareness and casual interactions did not provide rich awareness of others within the *context* of the community. My goal for this problem

foreshadowed the development of the ***Notification Collage Board,*** a tool that appropriately situated informal awareness cues and casual interactions of an entire community in a shared public space (see chapters 3 and 4).

The NC board is now a successful system that enjoys considerable use and comment. Indeed, it is the most used groupware system in our research group. It bridged distance for distributed community members, allowing them to stay in touch from distant locations via the internet. Several research projects have spun off locally from this project already, and future work in this research stream has already begun to be addressed.

5. The second problem commented on ***closed systems.*** I stated that present-day tools were closed systems offering a restricted set of channels for informal awareness and casual interactions. It was difficult or impossible to rapidly extend them to afford new kinds of rich, multimedia-based interactions required by the community. My goal for addressing this problem was the development of a ***Media Item Architecture*** (see chapter 5)**.** The Notification Collage Board was architected so that I and other developers could quickly and simply extend it with media items that afforded novel multimedia-based awareness and interactions.

   The architecture effectively separated the development of media items from the development of the NC board while keeping the two pieces in a single user interface. Also, it allowed people using the NC board to immediately take advantage of newly developed media items with very little overhead. The architecture was then given out to a class of undergraduate students, who successfully developed several novel media items with no prior knowledge of the system. The shared dictionary server system also proved to be a very useful and successful model to develop against, and the data model was exploited by other researchers to create additional tools for users of the NC board.

6. My third research problem highlighted ***impoverished interactions in the community.*** Present-day tools for informal awareness and casual interactions often provided just one channel for awareness and one channel for interactivity. Rich channels were

available separately, but there was no one tool that brought them all together *and* situated them within the community. My solution to this problem was the development of ***media items.*** We built a stock set of media items that provide rich channels for informal awareness and casual interaction. These items plug into the media item architecture so that they can appear on the NC Board as part of the community shared interaction space (see chapter 4). Further to this, new media items are still being explored, with the goal of further supporting a bridge to real time conversation and work.

6.4 Contributions

In this thesis, I contributed an interface and an architectural model for developing multimedia groupware that supports informal awareness and casual interaction in a community of intimate collaborators. I also contributed a new way of thinking about groupware, where co-located and distributed participants are both supported via public and private displays. I showed how the Notification Collage was successful in maintaining social bonds of intimate collaborators over distance. I also briefly mentioned how this research has directly and indirectly influenced new projects both inside and outside of our laboratory.

I also contribute the idea of media items and how they should be deployable on the fly. The use of media items in a universal public collaborative space makes exploration of novel groupware designs straightforward and quick, with the added advantage that a pool of users is there to immediately begin using and evaluating newly developed items.

Finally, I contribute a practical system. As this thesis is written, the Notification Collage board is actively used by a community for daily casual interactions.

References

1. Bly, S.A., Harrison, S.R., and Irwin, S. (1993), **Media Spaces: Bringing People Together in a Video, Audio, and Computing Environment**, *in Communications of the ACM, ACM Press, vol. 3, no. 1, pp. 28 47.*

2. Boyle, M. and Greenberg, S. (2002) **GroupLab Collabrary: A Toolkit for Multimedia Groupware**. *In J. Patterson (Ed.) ACM CSCW 2002 Workshop on Network Services for Groupware, November.*

3. Cadiz J.,Venolia G., Jancke G., Gupta A. (2002) **Designing and deploying an information awareness interface** *Proceedings of ACM CSCW'02 Conference on Computer-Supported Cooperative Work 2002 p.314-323*

4. Churchill, E., Nelson, L., Denoue, L., Girgensohn, A. (2003) **The Plasma Poster Network: Posting Multimedia Content in Public Places**. *Submitted for publication at INTERACT 2003, Ninth IFIP TC13 International Conference on Human Computer Interaction, Zurich, Switzerland, 1 September, 2003.*

5. Clark and Brennan (1991) **Grounding in Communication**. *P222-234, Text.*

6. Curtis, P., Nichols, D. A. (1993) **MUDs grow up: Social virtual reality in the real world**. *In Proceedings of the Third International Conference on Cyberspace, May 1993*

7. Denoue, L., Nelson, L., Churchill, E. (2003) **AttrActive windows: dynamic windows for digital bulletin boards** *Short talks-Specialized section: peripheral and*

ambient displays. Proceedings of ACM CHI 2003 Conference on Human Factors in Computing Systems 2003 v.2 p.746-747

8. Dourish, P., and Bly, S. (1992), **Portholes: Supporting Awareness in a Distributed Work Group**, *in Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing Systems (CHI'92), Monteray, CA, pp. 541 547.*

9. Fass, A., Forlizzi, J., Pausch, R. (2002) **MessyDesk and MessyBoard: two designs inspired by the goal of improving human memory.** *In proc of Designing Interactive Systems (DIS) 2002, 303-311*

10. Fish, R.S., Kraut, R.E., and Chalfonte, B.L. (1990), **The VideoWindow System in Informal Communications**, *in Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'90), Los Angeles, pp. 1 11.*

11. Fish, R.S., Kraut, R.E., Rice, R.E., and Root, R.W. (1993), **Video as a Technology for Informal Communication.** *In Communications of the ACM, ACM Press, vol. 36, no. 1, pp. 48 61.*

12. Fitzpatrick, G., Kaplan, S., Mansfield, T., Arnold, D., Segall, B. (2002) **Supporting Public Availability and Accessibility with Elvin: Experiences and Reflections**. *Computer Supported Cooperative Work 2002 v.11 n.3/4 p.447-474*

13. Greenberg, S. and Bohnet, R. (1991). **GroupSketch: A multi-user sketchpad for geographically-distributed small groups.** *In Proceedings of Graphics Interface '91, p207-215, Calgary, Alberta, June 5-7, Morgan-Kaufmann.*

14. Greenberg, S. (1996), **Peepholes: Low Cost Awareness of One's Community**. *In Companion Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing Systems (CHI'96), Vancouver, pp. 206 207.*

15. Greenberg, S. and Kuzuoka, H. (1999). **Bootstrapping Intimate Collaborators.** *In OzCHI99 Workshop: Issues of Use in CSCW Technology Design (held as part of the OZCHI'99 Australian Conference on Computer Human Interaction). Organized by Robertson, T., Fitzpatrick, G. and Greenberg, S., November 27, Wagga Wagga Australia.*

16. Greenberg, S. and Roseman, M. (1999). **Groupware Toolkits for Synchronous Work**. *In M. Beaudouin-Lafon, editor, Computer-Supported Cooperative Work (Trends in Software 7), Chapter 6, p135-168, John Wiley & Sons Ltd, ISBN 0471 96736 X. 258pp*

17. Greenberg, S. and Rounding, M. (2001) **The Notification Collage: Posting Information to Public and Personal Displays.** *Proceedings of the ACM Conference on Human Factors in Computing Systems [CHI Letters 3(1)], 515-521, ACM Press.*

18. Grudin, J. (2001) **Partitioning Digital Worlds: Focal and Peripheral Awareness in Multiple Monitor Use.** *In Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing Systems (CHI2001), Seattle, pp. 458-465*

19. Huang, E., Mynatt, E. (2003) **Semi-public displays for small, co-located groups** *Proceedings of ACM CHI 2003 Conference on Human Factors in Computing Systems 2003 v.1 p.49-56*

20. Jancke, G., Venolia, G. D., Grudin, J., Cadiz, J. J., Gupta, A. (2001) **Linking Public Spaces: Technical and Social Issues.** *Proceedings of ACM CHI 2001 Conference on Human Factors in Computing Systems 2001 p.530-537*

21. Kerne, A. (1997) **CollageMachine: Temporality and Indeterminacy in Media Browsing via Interface Ecology** *Short Talks: Browsing and Navigation, Proceedings of ACM CHI 97 Conference on Human Factors in Computing Systems 1997 v.2 p.297-298*

22. Kraut, R. E., Fish, R.S., Root, R.W., Chalfonte, B.L. (1990), **Informal Communication in Organizations: Form, Function, and Technology**, *in Peoples Reactions To Technology, Oskamp and Spacapan eds., Sage Publications, pp. 145 199.*

23. Kraut, R. E., Egido, C., and Galegher, J. (1988), **Patterns of Contact and Communication in Scientific Research Collaboration,** *In Proceedings of Computer Supported Cooperative Work (CSCW'88), New York, pp. 1 12.*

24. Mantei, M.M., Baecker, R.M., Sellen, A.J., Buxton, W.A.S., Milligan, T., and Wellman, B. (1991), **Experiences in the Use of a Media Space,** *in Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing Systems (CHI'91), New Orleans, pp. 203 208.*

25. Nardi, B.A., Whittaker, S., and Bradner, E. (2000), **Interaction and Outeraction: Instant Messaging in Action**, *in Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'00), Philadelphia, pp. 79 89.*

26. Patterson, J., Day, M. and Kucan, J. (1996) **Notification Servers for Synchronous Groupware**. *Proceedings of ACM CSCW'96 Conference on Computer-Supported Cooperative Work p. 122-129, ACM Press.*

27. Roseman, M. and Greenberg, S.(1996) **TeamRooms: Groupware for Shared Electronic Spaces.** *ACM SIGCHI'96 Conference on Human Factors in Computing System, Companion Proceedings, p275-276, ACM Press.*

28. Roseman, M. and Greenberg, S. (1997). **Simplifying Component Development in an Integrated Groupware Environment.** *Proceedings of the ACM UIST'97 Symposium on User Interface Software and Technology, p65-72, October 14-17, Banff, Alberta. ACM Press.*

29. Rounding, M. and Greenberg, S. (2000) **Using the Notification Collage for Casual Interaction**. *ACM CSCW 2000: Position Paper for the Workshop on Shared Environments to Support Face-to-Face Collaboration. Philadelphia, Pennsylvania, USA, December.*

30. Stefik, M., Bobrow, D., Foster, G., Lanning, S. and Tatar, D. (1986) **WYSIWIS Revised: Early Experiences with Multiuser Interfaces**, *in Readings in Groupware and Computer-Supported Cooperative Work, R. Baecker, Ed. Proceedings of the 1986 ACM conference on Computer-supported cooperative work*

31. Tang, C. (2003) **Capturing and Visualizing Histories of Multimedia-based Casual Interactions.** *M.Sc. Thesis, Department of Computer Science, University of Calgary, Calgary, Alberta CANADA. December.*

32. Whittaker, S., Frohlich, D., and Daly-Jones, O. (1994), **Informal workaplace communication: What is it like and how might we support it?,** *in Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing Systems (CHI'94), Boston, pp. 131 137.*

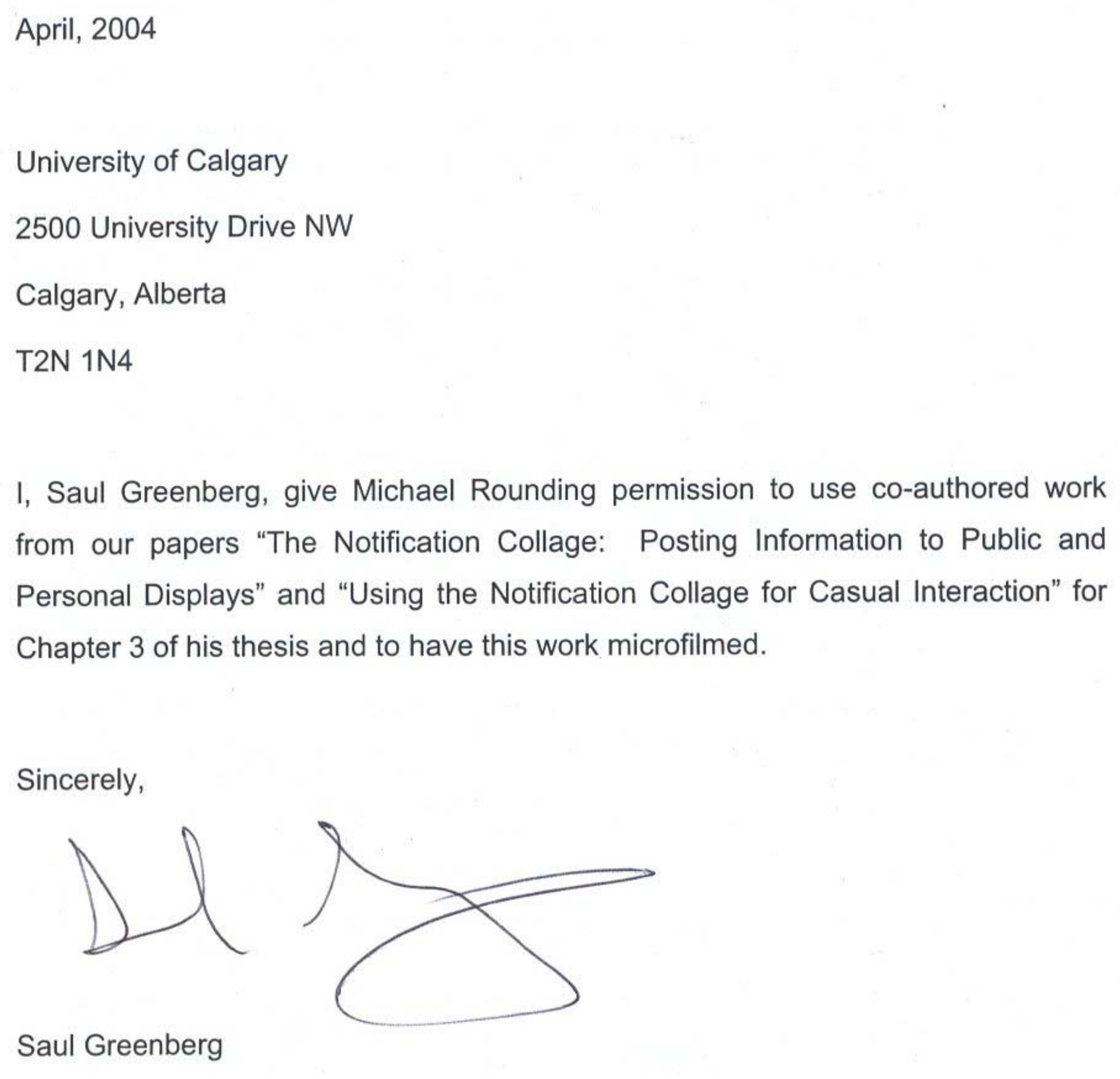
Appendix A. Table of Acronyms

This is an alphabetical list of acronyms used in this thesis and what they represent.

Table 2 - List of Acronyms

API	Application Programming Interface
COM	Component Object Model
DLL	Dynamic Link Library
GUID	Globally Unique Identifier
GUI / UI	Graphical User Interface / User Interface
IM	Instant Messaging
MIME	Multipurpose Internet Mail Extensions
MUD	Multi-User Dungeon
MVC	Model View Controller data model
NC	Notification Collage
NSTP	Notification Service Transfer Protocol
ProgID	Program Identifier
SD	Shared Dictionary
SDK	Software Development Kit
URL	Uniform Resource Locator
VB6	(Microsoft) Visual Basic 6

Appendix B. Co-Author Permission



April, 2004

University of Calgary

2500 University Drive NW

Calgary, Alberta

T2N 1N4

I, Saul Greenberg, give Michael Rounding permission to use co-authored work from our papers "The Notification Collage: Posting Information to Public and Personal Displays" and "Using the Notification Collage for Casual Interaction" for Chapter 3 of his thesis and to have this work microfilmed.

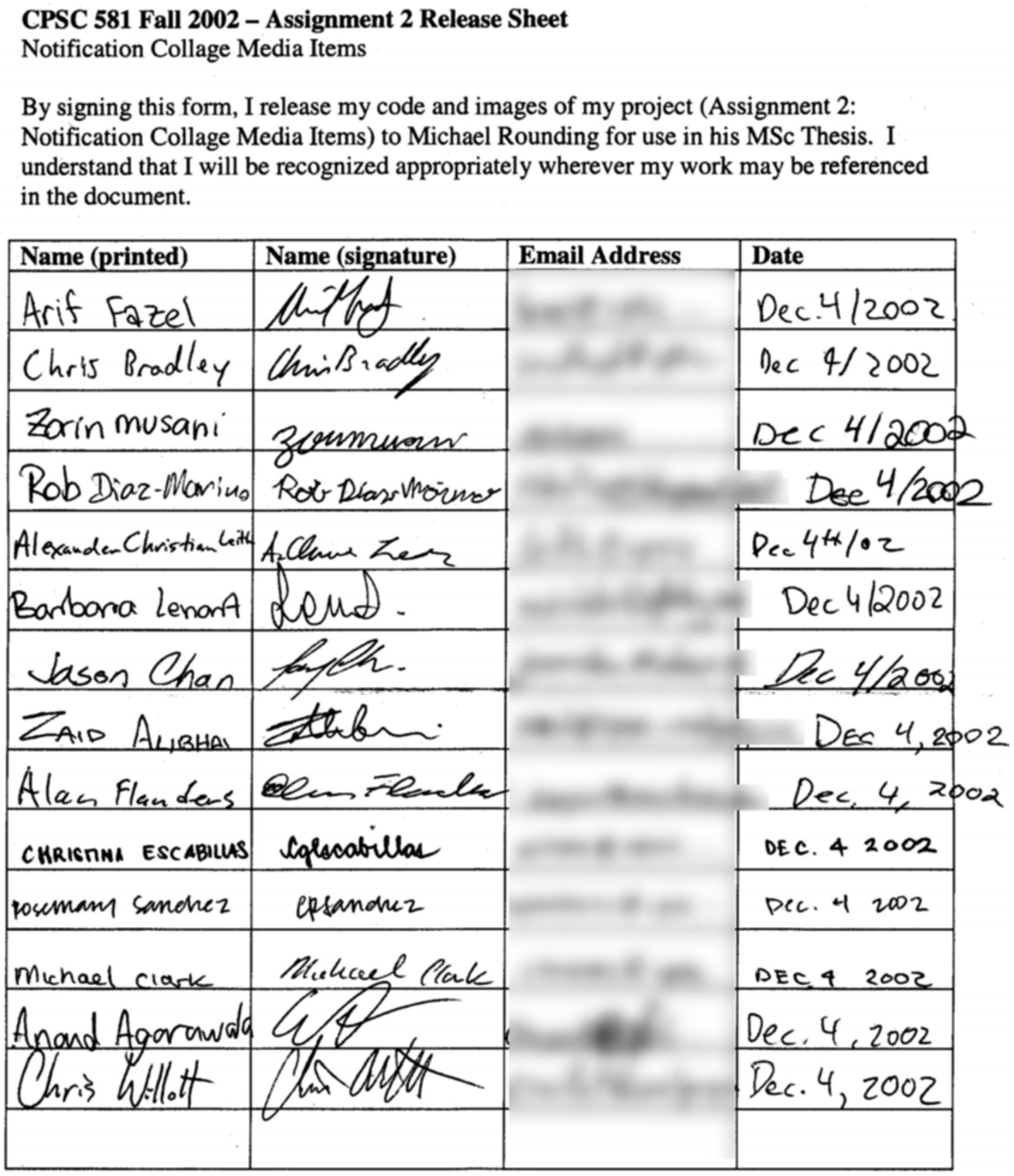
Sincerely,

[illegible]

Saul Greenberg

Appendix C. Student Release Form

Email addresses of students blurred to protect the privacy of the individuals.



CPSC 581 Fall 2002 – Assignment 2 Release Sheet
Notification Collage Media Items

By signing this form, I release my code and images of my project (Assignment 2: Notification Collage Media Items) to Michael Rounding for use in his MSc Thesis. I understand that I will be recognized appropriately wherever my work may be referenced in the document.

Name (printed)	Name (signature)	Email Address	Date
Arif Fazel	[illegible]	[redacted]	Dec.4/2002
Chris Bradley	[illegible]	[redacted]	Dec 4/2002
Zorin musani	[illegible]	[redacted]	Dec 4/2002
Rob Diaz-Marino	[illegible]	[redacted]	Dec 4/2002
Alexander Christian Leith	[illegible]	[redacted]	Dec 4th/02
Barbara Lenart	[illegible]	[redacted]	Dec 4/2002
Jason Chan	[illegible]	[redacted]	Dec 4/2002
ZAID ALIBHAI	[illegible]	[redacted]	Dec 4, 2002
Alan Flanders	[illegible]	[redacted]	Dec. 4, 2002
CHRISTINA ESCABILLAS	[illegible]	[redacted]	DEC. 4 2002
Rosemary Sanchez	[illegible]	[redacted]	Dec. 4 2002
Michael Clark	Michael Clark	[redacted]	DEC 4 2002
Anand Agarawala	[illegible]	[redacted]	Dec. 4, 2002
Chris Willott	[illegible]	[redacted]	Dec. 4, 2002