Cite as:
Greenberg, S., Voida, S., Stehr, N. and Tee, K. (2010) Artifacts as Instant Messaging Buddies. 11th Persistent Conversation Minitrack, Digital Media and Content, Proc. Hawaii International Conference on System Sciences - HICSS'2010, (Kauai, Hawaii, January 5-8), IEEE Computer Society.

Artifacts as Instant Messaging Buddies

Saul Greenberg, Stephen Voida, Nathan Stehr and Kimberly Tee
Department of Computer Science, University of Calgary
Calgary, Alberta CANADA T2N 1N4
saul.greenberg@ucalgary.ca

Abstract

Interpersonal awareness is one person's up-to-the moment knowledge about other group members, while artifact awareness is one person's up-to-the-moment knowledge about other group members' work artifacts; both contribute to fluid group coordination and interaction. Many popular computer-mediated communication systems focus on interpersonal awareness, with artifact awareness being a second-class citizen. To better support artifact awareness in this domain, we leverage the widespread adoption and technical robustness of an unaltered, commercial instant messaging (IM) system. Our main contribution is to treat an artifact, such as a document, as a first-class instant messaging "buddy." Our system, Artifact Buddy, gives each shared artifact an interactive presence on IM, providing awareness cues about its editing status and notifications when new versions are committed, enabling simple version control, and facilitating informal group communication. The group's interactions with and conversation around the artifact are also recorded as part of a persistent conversational history.

1. Introduction

An important component of the awareness necessary for casual interaction is *artifact awareness*: one person's up-to-the-moment knowledge of the artifacts with which other group members are working (a related concept is *document awareness* in prior research [20]). Such artifacts include the documents and drawings that individual group members work on over the course of a day as they pursue their collective work. These artifacts effectively serve as *boundary objects* around which communication, collaboration, and shared work take place [29]. For example, Whittaker and colleagues [36] found that over half of all casual interactions in an office involved some form of document sharing, where documents were mostly used as a cue or conversational prop. As detailed in [16, 34, 36] and summarized by Tee [31], being aware of such artifacts is valuable for many reasons:

- *monitoring* the progress of other group members on a shared document;
- *coordinating* joint activities among the group;
- *triggering interest* by seeing another person's activity, even if it is not part of a joint task;
- *determining availability,* where knowledge of artifact usage suggests how busy people are and if they can be interrupted; and
- *creating serendipitous opportunities* for people to engage in artifact-oriented collaborations.

Artifact awareness is easy when people are co-located (primarily because of the artifact visibility), but is problematic for distributed groups [16]. Consequently, various technical solutions providing artifact awareness have been implemented (see [34] for examples). "Explicit push" occurs when people explicitly send artifacts to others as part of a dialog, such as file exchange via email or within an instant messaging chat session [34]. "Explicit pull" occurs when people retrieve files that were somehow made available by others, for example, with peer-to-peer file sharing, document repositories [3, 20, 34], and version control systems. "Artifact availability" happens when people post artifacts through some kind of awareness server that displays the artifact state, e.g., OpenMessenger [4], Notification Collage [13], Community Bar [31], and Sharing Palette [34].

While these later systems provide a fine granularity of artifact awareness [13, 31, 34], they necessitate the use of specialized software. This requirement presents a number of problems. First, each additional system introduced onto the computer desktop is yet another awareness mechanism that competes for attention. Second, these systems are usually artifact specific; they rely on specialized software that is closely integrated with the artifact editor. As a result, generalizability across multiple document types and critical mass are difficult to achieve. Third, many systems dissociate artifact awareness from the interpersonal awareness of group members. This conflicts with observations from

everyday life, where artifact awareness is closely associated with interpersonal awareness of the people who share these artifacts [4, 34, 36].

Instead, we believe we can incorporate artifact awareness within an existing, highly used, and unaltered commercial awareness system: instant messaging (IM). IM allows ad hoc groups of friends (buddies) to maintain awareness of one another's availability and activity, and it facilitates easy transitions from passive awareness into conversation and interaction. IM is already extensively used in the workplace as an interpersonal awareness tool, and its benefits in this domain have been widely studied and discussed in the literature (e.g., [15, 22, 24, 30, 33]).

In this paper, we present a system called Artifact Buddy, which is grounded on the premise that an unaltered IM can simultaneously provide both artifact awareness and interpersonal awareness. In Artifact Buddy, artifacts become a first-class IM buddy and behave like other buddies within a defined group. The artifact-as-buddy knows which people are interested in it and notifies these individuals about its state. Group members can interact with the artifact (and the rest of the group) through the IM system's standard chat features.

Critically, this is all done with an existing and unaltered IM system. The IM system does all the heavy lifting: it does the underlying distributed systems work, communication, account control, and so on. For a group that already uses this common IM program, all that is required is that one group member install a helper application to run in the background. Additionally, because our approach takes advantage of the interaction mechanisms already well established by IM, group members can readily join and participate in collaborations without requiring that they learn how to use a completely new application.

We built Artifact Buddy as a working technical illustration of how artifact awareness can be feasibly integrated into an existing instant messenger. Our goal was not to build a more generally deployable system; this would require further (but routine) iterative development and testing of its user interface. We also believe that Artifact Buddy, as described below, is just one of many ways we can integrate artifact awareness within instant messaging.

2. Artifact Buddy

The Artifact Buddy system implements a user interface (annotated in Figure 1) and a wrapper around Microsoft's Live™ Messenger service [19]. We chose Live Messenger because it has functions typical of most IM services, as well as a public API; we use the open-source DotMSN library [38] to access Live

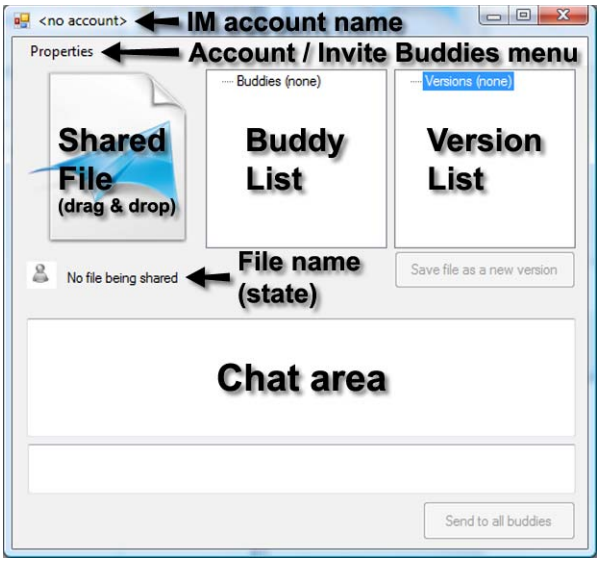

Figure 1. The (annotated) Artifact Buddy as it first appears.

Messenger functions. Through this API, Artifact Buddy programmatically invokes activities such as inviting buddies, setting and receiving state information, sending and receiving chat messages, initiating and responding to file exchanges, and so on. Importantly, Artifact Buddy is not a distributed system. Rather, it is a local application that relies completely on the underlying capabilities of the Live Messenger IM infrastructure to connect and to distribute chat data, status messages and files to others.

2.1. Using Artifact Buddy: A Scenario

We explain the features of Artifact Buddy by scenario. Mary is working on a conference paper with her co-authors Kim and Nathan, each of whom use Live Messenger as they normally would. As the primary author, Mary keeps the "master" copy of the paper, and coordinates with Kim and Nathan over changes and updates. Because she holds the master, she is the only one that initially needs the Artifact Buddy software, and the only one that has to carry out a few minor housekeeping chores in order to get things going.

2.1.1. Creating an IM Account for an Artifact. Mary first creates an IM buddy account for the artifact. She starts up her Windows Live Messenger client and, using its standard interface, signs up for a new account ID (identified by a made-up email address) to represent document(s) relevant to her group. She calls it "ConfPapers@live.com." Of course, the IM infrastructure has no idea that this account represents a document, rather than a real person.

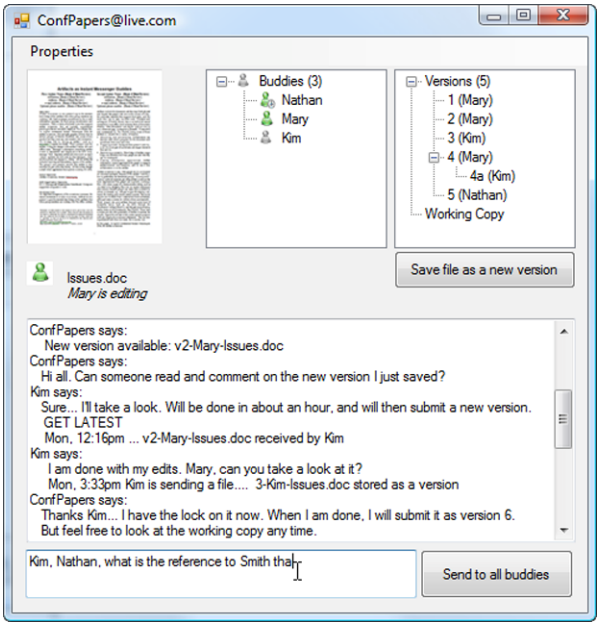

Figure 2. Artifact Buddy after a period of use.

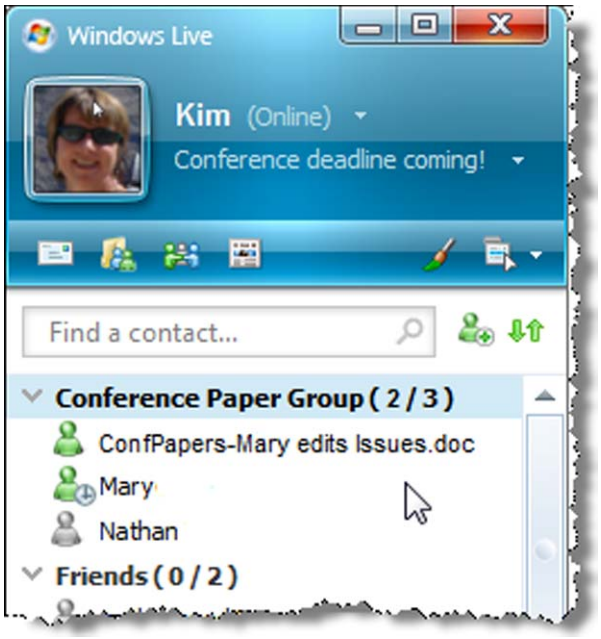

Figure 3. Kim's IM window showing ConfPapers.

2.1.2. Linking Artifact Buddy to Windows Live Messenger. Next, Mary needs to link Artifact Buddy with that IM account she just created. She starts Artifact Buddy, which appears on her screen as annotated in Figure 1. Using the functions accessible through the "Properties" menu (top left), she provides the just-created Live Messenger account ID and password. Using this information, Artifact Buddy can now link into Live Messenger by logging onto that account through the Messenger API.

2.1.3. Inviting Buddies. She now invites buddies—that is, people who are interested in the document being shared—to form a group. Specifically, she provides Kim's, Nathan's and her own Live Messenger IDs (their email addresses) to Artifact Buddy, again via the Properties menu. Internally, Artifact Buddy uses the Messenger API to send the invitation to each of them and then monitor their state. As each invitee accepts the invitation, their names will appear in the Artifact Buddy's "Buddy List" pane, along with an icon indicating each group member's on-line, idle and off-line state (Figures 1 and 2, top middle). Following the norms of Live Messenger, the Artifact Buddy account name (shortened as a nickname) also appears as a new contact in each person's Live Messenger buddy list, i.e., "ConfPapers." Figure 3 illustrates this in Kim's view, where, using Live Messenger's built-in grouping feature, she has created a new group within Live Messenger called "Conference Paper Group" and moved Mary, Nathan and the "ConfPapers" artifact-oriented buddy into it.

As an alternative to the artifact inviting people, others could instead use the standard IM features to invite the artifact to be their buddy (i.e., by inviting ConfPapers@live.com). Currently, Artifact Buddy is configured to accept all such invitations; this access could also be restricted to a "pre-approved" set of collaborators by comparing requests against an access control list, or requiring authorization by Mary.

2.1.4. Sharing an Artifact. Next, Mary drags and drops the artifact to be shared—in this case a word document titled "Issues.doc"—onto the Shared File icon (Figure 1, top left). Artifact Buddy immediately creates and stores a copy of this file as version 1. It displays the file as a thumbnail (Figure 2, top left) and as an item in the "Version List" pane, identified both by its version number and the person who submitted it (Figures 1 and 2, top right). Internally, this version copy is stored in a local directory called "versions," where the file name is prefixed with the version number and its creator, e.g., v1-Mary-Issues.doc.

The most current copy of the file is also kept by Artifact Buddy as the "working document" and listed in this pane. This working document is the one that Mary (and others) normally edit; it does not become a new version until it is submitted as such, a process which we explain in subsequent sections.

These first three steps are only required the first time that the Artifact Buddy is used to establish a

collaboration among a specific group. If the Artifact Buddy application is ever closed, Mary can simply re-start it and the associated Messenger account, list of buddies, and complete version history of the shared file are all automatically used to restore the prior working state of the collaboration. As an aside, Mary can reuse this account with different documents, although this would typically occur only after the group had completed its work on the previous one.

2.1.5. Maintaining Basic Artifact Awareness. Mary now begins to work on this document as she normally would. Other people see her activity in their standard Live Messenger client, as illustrated in Figure 3. They see that the artifact-as-buddy is online (the green buddy icon). They also see additional detail in the display name field associated with the buddy. As Mary makes changes, the display name changes to say she is editing the file (as illustrated in Figure 3). If she pauses editing or closes the document, the icon changes to reflect its "away" state. If she shuts down Artifact Buddy or logs off her computer, it is displayed as being off-line. For each case, the display name message field changes accordingly to provide detail. This behavior is inspired by previous document awareness tools based on the metaphors established by IM [20] and studies of interpersonal awareness practices in IM, where the manipulation of one's display name is used to indicate more detail about ongoing activities and availability [28].

Behind the scenes, Artifact Buddy monitors whether the shared file is opened by periodically examining the names of open windows on the computer where the Artifact Buddy application is being executed. If the file is discovered to be open in a window, Artifact Buddy uses the Windows Hooks API to monitor keyboard activity in that window to detect if the file is being actively edited. This information is translated into state and display name messages and subsequently transmitted through the Live Messenger API.

2.1.6. Saving Versions. At any point, Mary can specify that a copy of the working file should be saved as a new version (the "Save file as new version" button shown in Figure 2, middle right). Internally, Artifact Buddy implements a lightweight—albeit simplistic—version management system. It copies the file to the local artifact storage folder, numbers the new version, and tracks who submitted it. That version and the name of its submitter then appear on the Version list (Figures 1 and 2, top right). At any time, Mary can open previous versions by double clicking on them in the list.

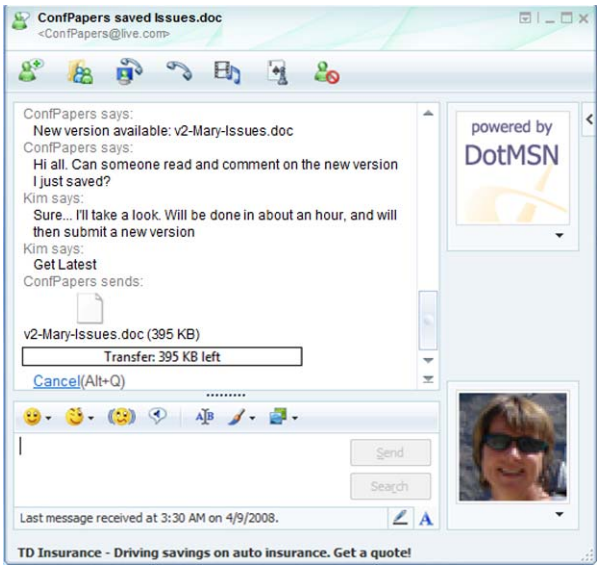

Figure 4. A sample of conversations, commands, and events as they appear in the IM chat dialog window.

Using a conversation-based interaction technique that we will present in the next section, other buddies can retrieve different versions of the document using variants of the `Get` command. They can also submit new revisions by dragging and dropping a file into their own IM chat dialog box. When a previous version is resubmitted, it is saved as a new version. That is, if version 4 of the shared file was edited and then resubmitted, it is saved as version 5. However, if someone else had submitted a version in the meantime, the system parses the artifact's filename and stores it as a child of the version from which it derives. For example, the version tree shown in Figure 2 (middle right) shows that Kim has retrieved, edited and resubmitted version 4 of the document. Since Nathan had submitted version 5 in the meantime, Kim's contribution is stored and listed as version 4a.

When a version is saved, Artifact Buddy initiates a chat dialog with all other buddies. For each buddy, a standard Live Messenger chat box appears, illustrated by the dialog in Figure 4. Artifact Buddy sends a message indicating that a new version of a file is available. For certain types of files, it also summarizes the differences between the current and previous version. These machine-generated messages are also recorded in the Artifact Buddy application's chat area (Figures 1 and 2).

2.1.7. Conversing with Artifacts. Any of the group members can initiate a dialog directly with the artifact by chatting with it, i.e., by typing commands into the standard IM chat box. For example, if Kim types `Get`

`latest` into the chat box, Artifact Buddy will respond by initiating a file transfer of the latest version of the paper, just as if the Artifact Buddy were another person and had dragged and dropped the file into the chat area (see Figures 2 and 4). Similarly, any buddy can issue any of the following commands:

- **`Versions`** lists all available versions;
- **`Get #`** gets a particular version identified by its number;
- **`Get working`** gets the current working copy of the file;
- **`Get original`** gets a copy of the original document, i.e., it is the same as `Get 1`;
- **`Get latest`** gets the latest saved version of the document;
- **`Get transcript`** gets a copy of the entire dialog entered by people as they converse with the artifact and with each other, as well as the events generated by the artifact; and
- **`Help`** displays a list of all commands that the artifact understands.

All submitted commands are also displayed in the Artifact Buddy chat area (Figures 1 and 2, bottom) and relayed to the chat dialogs of other buddies (Figure 4). This persistent visualization of all group members' interactions with the artifact provides each of the collaborators with ongoing awareness of all activity concerning the shared file and its various versions.

2.1.8. Conversing with the group. Any group member can converse not only with the artifact, but also with all buddies associated with that artifact. If the text entered in an IM chat with the artifact-as-buddy is not recognized as a command, Artifact Buddy assumes it is a message intended for the group and re-broadcasts it to all participants. Similarly, the master—in this case, Mary—can use the Artifact Buddy chat window to broadcast messages to all other buddies. We can now see that the (different) dialogs shown in Figures 2 and 4 contain a mix of commands, events, interpersonal dialog, and file transfers. In this way, conversations with the artifact serve both as a command mechanism and a channel for multi-party discussions around the artifact, without requiring the use of multiple applications—or even multiple IM chat windows.

2.1.9. Accessing a Persistent Record of Interactions with and Conversation around the Artifact. Because group members' interactions with the shared artifact take place in the context of an IM conversation, it is possible to scroll back in the conversation history to gain a quick overview of the activity of the group. As long as each person's Artifact Buddy IM conversation window is left open, this information will continue to accumulate as commands are carried out and conversations held around the shared artifact over time. Recent versions of Windows Live Messenger will also allow these conversations to be stored from session to session, enabling group members to construct long-running histories of the group's collaborative work.

Additionally, Artifact Buddy retains a persistent transcript of all messages, stored locally in an XML file. At any time, any person can request and review this transcript of events, of submitted commands, and of the group dialog around the artifact by issuing the **`Get transcript`** command. This ability to review the transcript is especially important for people who have been offline for a while, as it helps them get up-to-date with any activity that has taken place in their absence [7, 11].

2.2. Sharing Artifact Control

In order to minimize the collaboration overhead for the individual members of the group, our first implementation of Artifact Buddy required only one member of a group to install special software on their computer. Other group members could simply use their standard IM clients to participate in the collaboration. Although any member of the group could contribute new versions of an artifact by dragging and dropping updated versions into their own IM chat window, the artifact-as-buddy's status and display name would change to reflect live editing taking place in the master document on the computer running the Artifact Buddy software and serving as the "host" of the collaboration session. Because the other members of the group were not running custom software that could actively track interactions with the shared artifact, their editing actions were not reflected to the rest of the group until they explicitly created a new version of the document based on their changes. This arrangement worked reasonably well when one member of the group was serving as the primary document editor or the coordinator within the group.

However, in some collaborations, the hosting/coordinating role may not be fixed and may need to fluidly pass from one member of the group to another during different phases of the project. Additionally, with the original Artifact Buddy implementation, the artifact's online presence and availability to the rest of the group is dependent on the host's computer being turned on, logged in, and connected to the network. If the hosting contributor goes off-line, the artifact awareness sharing mechanism within the group is lost, and group members are limited

to working only with versions of the shared artifact that they had previously retrieved and cached locally on their own computers.

To support more flexible role assignment within collaboration groups, we created a second iteration of the Artifact Buddy software, adding three new commands:

- **`Announce editing`** manually changes the artifact's online status and sets its name and status messages to indicate that a particular group member is currently editing the artifact;
- **`Announce edit done`** posts a notification to the group that a prior editing session has completed and reverts the online status, name, and status messages to their normal behavior; and
- **`Request control`** initiates the download of an archive file containing the Artifact Buddy client software and the current artifact repository, allowing another collaborator to assume control over the master document and automatically issue status updates when they edit the document.

2.2.1. Distributing Control of Artifact Status. In some instances, group members may want to take control of the online status and display name of the artifact-as-buddy to communicate artifact awareness without having to fully commit to running the Artifact Buddy software on their own computer and hosting the master document. This capability would be particularly valuable in collaborations where multiple group members are contributing substantive edits to the shared document and want to prevent inadvertent introduction of overlapping or conflicting edits to the document.

If a group member issues the `Announce editing` command via the IM chat box, the online status of the artifact-as-buddy is set to online (the green buddy icon) and its display name is changed to communicate that the artifact is currently being edited, and by whom. At this point, the online status and display name of the artifact-as-buddy are frozen until one of the following occurs:

- the group member releases control of the buddy's status by issuing a `Announce edit done` command in their IM chat box,
- another group member sends an `Announce editing` command,
- the hosting collaborator begins to edit the document on their own machine (in which case their editing status overrides the other group members'), or
- the hosting collaborator logs off, suspending the collaboration session.

2.2.2. Transferring Control of the Artifact-as-buddy Among Collaborators. Any member of the group can also take complete control of the master document and host the artifact-as-buddy on his or her own computer. This transfer can happen in one of two ways. If the group member wishing to take control already has the Artifact Buddy software installed on their own computer, they can negotiate with the other group members about which version of the document should be the current working version, `Get` a copy of that version through their existing dialog with the artifact, start up their own instance of Artifact Buddy, and re-share the working version using the artifact's IM account. The drawback to this approach is that in transferring just one version of the artifact as a "working" version between hosts, the complete revision history and transcript of discussion and interaction around the artifact is lost.

With the new version of the system, a member of the group can issue the `Request control` command in their IM chat window. When this command is received, the current host is prompted to approve transferring control and hosting responsibilities to the person who made the request. If the request is approved, a ZIP archive containing the Artifact Buddy software is sent to the person requesting control. This software package automatically embeds all of the current session information, encrypted credential information to log into the artifact-as-buddy IM account, and revision history for the shared artifact. When the new host unzips and starts the Artifact Buddy software, the session is resumed in exactly the same state it was in prior to the control transfer, and the previous host is logged out. The control transfer is logged in the running transcript, and to the rest of the group members, the only disruption caused by the transfer of control is a quick log-out/log-in sequence by the artifact-as-buddy.

3. Discussion

Artifact Buddy illustrates a unique point in the design space of collaboration technologies by leveraging the capabilities of an existing computer-mediated communication platform to provide artifact awareness within collaboration groups. In this section, we discuss some of the specific contributions of the Artifact Buddy approach, reflect on the pros and cons of embedding artifact awareness in the context of a computer-mediated communication tool, and explore some of the tensions that Artifact Buddy uncovers in the design of collaboration "hubs."

3.1. Artifact Awareness using Computer-Mediated Communication Tools

A variety of mechanisms exist to streamline the process of sharing files across multiple machines, such as Dropbox [6], MobileMe [1], and SugarSync [26]. These kinds of collaboration tools generally focus on keeping files in sync across multiple computers, be they owned by a single person or by multiple members of a group. While maintaining consistent artifact state across multiple collaborators is an important part of facilitating artifact awareness, these artifact-focused implementations tend to separate the artifact sharing and communicative aspects of collaboration. These systems are very good at sharing *content*, but provide little or no support for communicating the *context* surrounding the artifacts; our approach focuses on artifact awareness and sharing in the context of computer-mediated conversations.

An approach more similar to ours is presented by Fitzpatrick et al. [9], who combined a notification infrastructure with a chat system in order to broadcast activity in a CVS repository to a group of developers who were using that repository to collaborate on a software project. While this system did foster artifact awareness in the context of conversations among the collaborators, it still maintained a separation between *working with* the shared artifacts (via the command-line or a CVS-aware development tool) and *conversing about* the shared artifacts (via the chat application).

With Artifact Buddy, the acts of sharing artifact state and working with the shared artifacts are both embedded within the communications channel provided by IM. There are a number of benefits that can be realized from incorporating artifact awareness and management into computer-mediated communication tools:

- Group members can utilize existing tools to carry out the collaboration, reducing the number of applications demanding their attention and helping to overcome the "critical mass" problem inherent in many groupware systems [14].
- IM systems provide "presence"-type awareness information as a cue for conversant availability, which can easily be extended to communicate the availability or editing state of a shared artifact [20].
- A complete and persistent record of all group conversation and interactions with the shared artifact(s) can be automatically captured and disseminated on demand to any participant [7, 11].

There are some drawbacks to this approach, however. Each collaboration group requires that a corresponding Windows Live Messenger account be created and maintained to represent the shared artifact and support the ongoing collaboration. Because the approach overloads the group chat channel to create a means for issuing commands to the shared artifact(s), constraints—albeit small ones—are imposed on the kinds of communicative messages that group members can send to one another. The Artifact Buddy implementation is also subject to limitations introduced by the underlying communications protocol, leading, for example, to an arbitrary ceiling on the size and file types of artifacts shared through the system.

Instant messaging is not the only computer-mediated communications channel that could be exploited (or augmented) to provide artifact awareness capabilities; there are a multitude of other services now available. These include web-based social networks (e.g., Facebook [8], MySpace [21]), micro-blogging systems (e.g., Twitter [32]), blogging systems (e.g., Moveable Type [27], WordPress [37]), and so on. Each has affordances built for interpersonal awareness that lend themselves to artifact awareness in ways somewhat similar to what we have done with instant messaging systems. For example, an artifact could register as a "person" on a social networking site, and update others by posting status updates or micro-blog entries. More detailed summaries could be posted as blog postings. Depending on the social network, images of the changing artifact over time can be posted as photos or as shared files. Each of these variations preserve the concept of integrating interpersonal and artifact awareness in a persistent conversational stream, and all that is needed to implement these variations of Artifact Buddy are public APIs for the various awareness services. We hope to examine some of these possibilities in future work.

3.2. IM as a Collaboration "Hub"

Multifaceted groupware tools are often designed to serve as a "hub" around which collaboration and communication take place and where awareness information is exchanged. Artifact Buddy integrates artifact awareness into an existing commercial and unmodified IM client, a design decision that establishes the IM client window—or the Artifact Buddy window, on the host contributor's computer—as the primary collaboration hub when participating in a group that has adopted the system. Other research efforts and commercial products have proposed alternative collaboration hubs that are constructed around different constituent technologies. We argue here that Artifact Buddy demonstrates the potential usefulness of IM as a collaboration hub and helps to highlight some of its unique affordances in this space.

E-mail is frequently cited as one of the primary collaboration hubs utilized in the workplace, due primarily to its ubiquity and flexibility [2, 34, 36]. While e-mail is a somewhat impoverished medium for communicating awareness information—group and artifact awareness updates are typically conveyed implicitly as a side effect of outgoing, communication-driven messages—it is still frequently appropriated to this end, along with file sharing and other collaborative tasks [34]. Several research projects have sought to take advantage of the fact that so much of the workday is spent working out of an e-mail client, introducing additional collaboration tools into this important application (e.g., [2]). However, email suffers from several limitations when serving as a collaboration hub, two significant ones being that e-mail is a strictly "push"-oriented communications medium [34] and that, for most information workers, e-mail is already quite an overloaded channel.

Other systems have adopted this view of e-mail-client-as-hub but incorporated complementary technologies to help facilitate a broader variety of awareness and on-demand sharing capabilities. For example, IBM's ActivityExplorer [12] combines standard e-mail / chat mechanisms with RSS feed-based tools. While this kind of hub does provide more powerful collaboration support, it suffers the drawback of requiring all participants to adopt a new (and separate) collaboration tool in order to fully take advantage of the benefits the system offers. This tradeoff is also present in the use of other specialized groupware systems, e.g., Groove [17], Notification Collage [13], and the Community Bar [31].

Shared filesystems and document repositories have also been utilized as collaboration hubs, placing a heavier focus on artifact sharing and incorporating communication and awareness tools as a secondary component of the system. As we discussed earlier, Fitzpatrick et al.'s use of a chat system to broadcast CVS activity [9] falls into this category of systems. WikiFolders also enables filesystem-based collaboration by enabling rich-text annotation of shared files [35], although this system communicates only implicit awareness based on the content of the annotations that group members attach to files and folders. On a more fine-grained scale, some approaches have even transformed individual documents into hubs for collaboration in an attempt to close the gap between where work is taking place and the tools used to foster communication and awareness among group members (e.g., Anchored Conversations system [5]). Empirical studies have demonstrated that collaboration using document- or file-based venues sometimes breaks down in the absence of awareness about how others are using the system [23]; this suggests that improving the integration between interpersonal- and artifact-awareness in these contexts would also be of value.

Instant messaging provides an interesting alternative for serving as the basis of collaboration hubs. IM clients are already in widespread use in the workplace [15, 22, 24, 30, 33]; they provide powerful, automatically updated, and persistently visible awareness cues; and can serve both as a just-in-time, "push"-based communication medium *and* an on-demand, "pull"-based medium in which persistent transcripts of conversations and interactions can be retrieved when needed [7, 11]. Other research efforts, such as Project View IM [10][24][25], have taken advantage of these affordances to enable more structured collaboration capabilities based on instant messaging systems, and we believe that Artifact Buddy demonstrates how this idea of IM-as-collaboration-hub can be extended even further with the addition of artifact awareness tools.

Finally, using a commercial IM system as a communication hub means that we can exploit the huge efforts and costs that have gone into building, supporting and maintaining such a system. While Artifact Buddy itself is a prototype, the IM system it uses is not. The IM infrastructure does the heavy lifting, which means only modest improvements need to be made within our Artifact Buddy prototype to make it robustly deployable and usable to the millions of people who already use IM on a daily basis.

4. Conclusions and Future Work

Artifact Buddy is a demonstration that realizes our basic idea: that artifact awareness can be managed through the standard and unaltered interpersonal awareness and communication capabilities of instant messaging clients and other common computer-mediated communication technologies. In this paper, we provide just one illustration of how artifact awareness can be integrated into an existing system such that:

- members of the group can benefit from shared artifact awareness *without requiring the use of additional software*;
- *artifact sharing and version control capabilities are integrated* into the same interface that is already used to provide interpersonal awareness and communication capabilities; and
- a *persistent history* is maintained of all *interactions with the shared artifact* and all of the *conversations surrounding it*.

Finally, we demonstrated that instant messaging is an interesting technology to consider for supporting artifact awareness because it provides a number of

compelling characteristics, including a robust messaging infrastructure, broad adoption in the workplace, real-time awareness cues that can convey the status of shared artifacts, and a persistent conversation structure that can capture both interactions with and discussions around shared artifacts. This last characteristic is particularly essential for helping transient participants to stay up-to-date with the flow of the collaboration over time.

The implementation of our concept can have many variations that go far beyond what we have detailed here. Some suggestions follow.

4.1. Document-level locking

Our current implementation of Artifact Buddy provides no explicit document-level locking capabilities. As different members of the group take turns reviewing, editing, and sharing revisions to an artifact, the system relies mainly upon communication among members of the group and existing social conventions to ensure that the artifact is not being edited concurrently and that conflicting changes are not introduced. For the kinds of small, tight-knit groups that Artifact Buddy is intended to support, the close integration of communication and artifact-sharing tools, the availability of a persistent transcript of the discussion surrounding the artifact, and some lightweight capabilities to control a status message associated with the artifact often provide a strong enough sense of artifact awareness to prevent serious problems in coordinating the group's efforts.

In contrast, for larger or less closely connected groups, more explicit document-level locking could be managed by having the Artifact Buddy track the state of who "owns" a particular version of the artifact. This system could then enforce locks, allowing only that person to submit a subsequent version of the shared artifact. Such a system could also provide greater control over branching versions, as is done in many full-fledged version control systems. Indeed, it should be straightforward for a system like Artifact Buddy to use an existing version control system as its underlying document management engine.

4.2. Multiple artifacts

Our implementation only manages one document at a time. It could be extended to handle multiple documents. For example, a variant of Artifact Buddy could track multiple files, where it provides awareness of their global state both as a collection and as details of particular files as they are being used (e.g., in the display name and/or the chat dialog). Group members would be able to submit and receive files individually or as a collection (e.g., as a ZIP archive). However, such an approach would introduce a considerable additional complexity to the Artifact Buddy interface and the system's chat-based command language.

4.3. Artifact awareness as streamed window snapshots

Tee et al. [31] advocated screen sharing as a method to display on-going changes to a file. Their idea was to transmit a miniature version of a screen or window's image to other people, which can then be visually tracked to see changes as they occur. This approach could also be implemented in an IM-based system. Many instant messaging clients now have the capabilities to open a video channel, e.g., for web cam conversations. This feature can be exploited by having a system like Artifact Buddy take periodic window snapshots of the artifact as it is being edited, and repackage and stream them on-the-fly as video frames over the IM video channel. It could use the standard IM API and video codec protocol to transmit these frames. Microsoft's SharedView is one example of a service that provides this kind of screen sharing functionality [18], but it currently supports only rudimentary chat and file sharing. We speculate that a system that combines the interpersonal- and artifact-awareness aspects of Artifact Buddy and the synchronous screen-sharing capabilities of SharedView would be a powerful tool for supporting a diversity of collaboration practices.

In summary, we believe that artifact awareness can exploit the standard interpersonal awareness and communication capabilities of social systems such as instant messaging. Artifact Buddy is just one working example to illustrate how this can be done; many other variations are possible.

5. Acknowledgements

This work was partially funded by the NSERC/iCORE/SMART Industrial Research Chair in Interactive Technologies and by NSERC's NECTAR Research Network and Discovery Grant programs.

6. References

[1] Apple, Inc., "MobileMe," http://www.apple.com/mobileme/, accessed 5 June 2009.

[2] Bellotti, N., N. Ducheneaut, M.A. Howard, and I.E. Smith, "Taking email to task: The design and evaluation of a task management centered email tool," *Proc. CHI 2003*, ACM Press, 2003, pp. 345–352.

[3] Bentley, R., T. Horstmann, and J. Trevor, "The World Wide Web as enabling technology for CSCW: The case of BSCW," *Computer Supported Cooperative Work 6*, 2–3, 1997, pp. 111–134.

[4] Birnholtz, J.P., C. Gutwin, G. Ramos, and M. Watson, "OpenMessenger: Gradual initiation of interaction for distributed workgroups," *Proc. CHI 2008*, ACM Press, 2008, pp. 103–106.

[5] Churchill, E.F., J. Trevor, S. Bly, L. Nelson, and D. Cubranic, "Anchored conversations: Chatting in the context of a document," *Proc. CHI 2000*, ACM Press, 2000, pp. 454–461.

[6] Dropbox, "Dropbox," http://www.getdropbox.com, accessed 5 June 2009.

[7] Erickson, T., D.N. Smith, W.A. Kellogg, M. Laff, J.T. Richards, and E. Bradner, "Socially translucent systems: Social proxies, persistent conversation, and the design of 'Babble'," *Proc. CHI 1999*, ACM Press, 1999, pp. 72–79.

[8] Facebook, "Facebook," http://www.facebook.com, accessed 5 June 2009.

[9] Fitzpatrick, G., P. Marshall, and A. Phillips, "CVS integration with notification and chat: Lightweight software team collaboration," *Proc. CSCW 2006*, ACM Press, 2006, pp. 49–58.

[10] Fussell, S.R., S. Kiesler, L.D. Setlock, and P. Scupelli, "Effects of instant messaging on the management of multiple project trajectories," *Proc. CHI 2004*, ACM Press, 2004, pp. 191–198.

[11] Gergle, D., D.R. Millen, R.E. Kraut, and S.R. Fussell, "Persistence matters: Making the most of chat in tightly-coupled work," *Proc. CHI 2004*, ACM Press, 2004, pp. 431–438.

[12] Geyer, W., J. Vogel, L. Cheng, and M. Muller, "Supporting activity-centric collaboration through peer-to-peer shared objects," *Proc. GROUP 2003*, ACM Press, 2003, pp. 115–124.

[13] Greenberg, S. and M. Rounding, "The Notification Collage: Posting information to public and personal displays," *Proc. CHI 2001*, ACM Press, 2001, pp. 515–521.

[14] Grudin, J., "Groupware and social dynamics: Eight challenges for developers," *Communications of the ACM 37*, 1, 1994, pp. 92–105.

[15] Isaacs, E., A. Walendowski, S. Whittaker, D.J. Schiano, and C. Kamm, "The character, functions, and styles of instant messaging in the workplace," *Proc. CSCW 2002*, ACM Press, 2002, pp. 11–20.

[16] Kraut, R.E., C. Egido, and J. Galegher, "Patterns of contact and communication in scientific research collaborations," In Galegher, J., R.E. Kraut, and C. Egido (Eds.), *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1990, pp. 149–171.

[17] Microsoft Corporation, "Microsoft Office Groove," http://office.microsoft.com/groove/, accessed 5 June 2009.

[18] Microsoft Corporation, "Microsoft SharedView," http://connect.microsoft.com/site/sitehome.aspx?SiteID=94, accessed 5 June 2009.

[19] Microsoft Corporation, "Microsoft Windows Live™ Messenger," http://messenger.live.com, accessed 5 June 2009.

[20] Morán, A.L., J. Favela, A.M. Martínez, and D. Decouchant, "Document presence notification services for collaborative writing," *Proc. Seventh International Workshop on Groupware*, IEEE Computer Society, 2001, pp. 125–133.

[21] MySpace, Inc., "MySpace," http://www.myspace.com, accessed 5 June 2009.

[22] Nardi, B.A., S. Whittaker, and E. Bradner, "Interaction and outeraction: Instant messaging in action," *Proc. CSCW 2000*, ACM Press, 2000, pp. 79–88.

[23] Rader, E., "Yours, mine and (not) ours: Social influences on group information repositories," *Proc. CHI 2009*, ACM Press, 2009, pp. 2095–2098.

[24] Scupelli, P., S.R. Fussell, S. Kiesler, P. Quinones, and G. Kusbit, "Juggling Work Among Multiple Projects and Partner," *Proc. HICSS 2007*, IEEE Computer Society, 2007, Article 77.

[25] Scupelli, P., S. Kiesler, S.R. Fussell, and C. Chen, "Project View IM: A tool for juggling multiple projects and teams," *Ext. Abstracts CHI 2005*, ACM Press, 2005, pp. 1773–1776.

[26] Sharpcast, Inc., "SugarSync," http://www.sugarsync.com, accessed 5 June 2009.

[27] Six Apart, Ltd., "Moveable Type," http://www.moveabletype.org, accessed 5 June 2009.

[28] Smale, S. and S. Greenberg, "Broadcasting information via display names in instant messaging," *Proc. GROUP 2005*, ACM Press, 2005, pp. 89–98.

[29] Star, S.L. and J.R. Griesemer, "Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907–39," *Social Studies of Science 19*, 3, 1989, pp. 387–420.

[30] Tang, J. and B. Begole, "Beyond instant messaging," *ACM Queue 1,* 8, 2003, pp. 28–37.

[31] Tee, K., S. Greenberg, and C. Gutwin, "Providing artifact awareness to a distributed group through screen sharing," *Proc. CSCW 2006*, ACM Press, 2006, pp. 99–108.

[32] Twitter, Inc., "Twitter," http://www.twitter.com, accessed 5 June 2009.

[33] Voida, A., W.C. Newstetter, and E.D. Mynatt, "When conventions collide: The tensions of instant messaging attributed," *Proc. CHI 2002*, ACM Press, 2002, pp. 187–194.

[34] Voida, S., W.K. Edwards, M.W. Newman, R.E. Grinter, and N. Ducheneaut, "Share and share alike: Exploring the user interface affordances of file sharing," *Proc. CHI 2006*, ACM Press, 2006, pp. 221–230.

[35] Voida, S. and S. Greenberg, "WikiFolders: Augmenting the display of folders to better convey the meaning of files," *Proc. CHI 1999*, ACM Press, 2009, pp. 1679–1682.

[36] Whittaker, S., D. Frohlich, and O. Daly-Jones, "Informal workplace communication: What is it like and how might we support it?" *Proc. CHI 1994*, ACM Press, 1994, pp. 131–138.

[37] WordPress.org, "WordPress," http://www.wordpress.org, accessed 5 June 2009.

[38] Xih Solutions, "DotMSN .NET Messaging Library," http://www.xihsolutions.net/dotmsn/, accessed 5 June 2009.