IMPROMPTU: A New Interaction Framework for Supporting Collaboration in Multiple Display Environments and Its Field Evaluation for Co-located Software Development

Jacob T. Biehl[1], William T. Baker[1], Brian P. Bailey[1], Desney S. Tan[2], Kori M. Inkpen[2], and Mary Czerwinski[2]

[1]Department of Computer Science
University of Illinois
Urbana, IL 61801, USA
{jtbiehl, wtbaker, bpbailey}@uiuc.edu

[2]Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{desney, kori, marycz}@microsoft.com

ABSTRACT

We present a new interaction framework for collaborating in multiple display environments (MDEs) and report results from a field study investigating its use in an authentic work setting. Our interaction framework, IMPROMPTU, allows users to share task information across displays via off-the-shelf applications, to jointly interact with information for focused problem solving and to place information on shared displays for discussion and reflection. Our framework also includes a lightweight interface for performing these and related actions. A three week field study of our framework was conducted in the domain of face-to-face group software development. Results show that teams utilized almost every feature of the framework in support of a wide range of development-related activities. The framework was used most to facilitate opportunistic collaboration involving task information. Teams reported wanting to continue using the framework as they found value in it overall.

Author Keywords

Multiple Display Environments, Group Work, Group Software Development, Field Study

ACM Classification Keywords

H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces – collaborative computing.

INTRODUCTION

A multiple display environment (MDE) is comprised of co-located personal (e.g. laptops) and shared devices (e.g. large displays) that are networked to form an integrated virtual workspace [33]. These environments offer many potential benefits for small workgroups, including the ability to place myriad information artifacts on shared displays for comparing, discussing, and reflecting on ideas; to jointly create and modify information to enhance focused problem solving or enable opportunistic collaboration; and to allow quick and seamless transitions between these work modes.

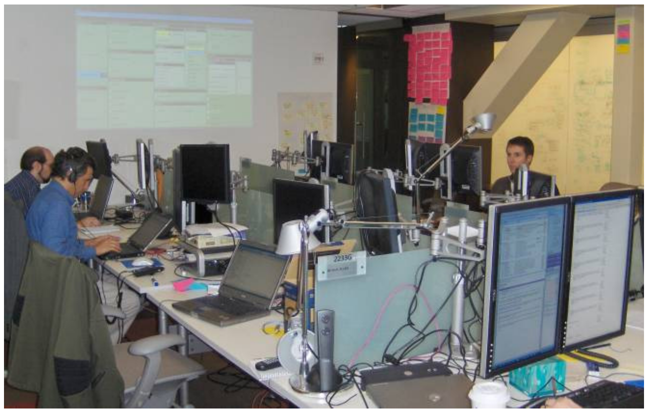


Figure 1: An example of a multiple device environment typically used for face-to-face software development.

One fundamental challenge is to build systems that allow groups to fully realize these benefits in MDEs. Many such systems have been investigated [19, 27, 34, 36], but have not adequately addressed this challenge. For example, with Colab [33], groups can only work with digital information supported by custom built applications. In the iRoom [19], relocating applications (information) does not maintain interaction context (e.g. stack traces in a debug window would be lost). This can hinder opportunistic collaboration.

A second challenge is to understand how groups leverage MDEs to perform their activities and the resulting impact. Many lab studies of such systems and related interactions have been conducted [25, 31, 33], but this corpus of work has not adequately answered the broader question of how MDEs are used for *real* activities in *authentic* settings.

Addressing both challenges, we present a new interaction framework for collaborating in MDEs and report results from one of the first field studies investigating how such frameworks are utilized in an authentic work setting. Our research was grounded in a particular task domain: face-to-face software development activities (see Fig. 1). This domain was chosen because it is representative of co-located group problem solving activities and the use of MDEs is a rapidly emerging practice in this domain [29].

Our IMPROMPTU framework allows groups to more fully realize core principles of group work in MDEs beyond what is provided by existing systems. Important features include the ability to share myriad information across displays via off-the-shelf applications without modification, to jointly interact with task information across devices for focused problem solving, and for different group members to place information on large displays for discussing, comparing, and reflecting on ideas. It also provides a lightweight interface for performing these and other related actions. The interaction design of our framework was based on lessons learned from our earlier experiences [5, 6] as well as surveys and interviews with users in our target domain.

We deployed our framework within authentic workspaces used by two software development teams and studied its use and impact over a three week period. Our results show that teams utilized almost every feature of the framework in support of a wide range of problem solving activities. The framework was used most often to facilitate opportunistic collaborations that involved task information. Teams reported wanting to continue using the framework as they found value in it overall. We also discovered new insights for how to improve the design of MDE frameworks.

RELATED WORK

We review principles of group work and how systems have sought to realize them for MDEs, the role of MDEs in emerging software development practices, and why existing tools are insufficient for supporting those practices.

Realizing Principles of Group Work within MDEs

Researchers have conducted many studies of group work to extract principles for building supporting systems [15]. Fundamental principles include the ability to create, share and exchange task information, to allow individual work in parallel and joint interaction, to allow seamless transitions between these work modes, and to maintain awareness of each other's activities (e.g. see [13, 16, 23, 30, 33]).

For multiple display environments (and other co-located workspaces), a challenge has been to understand how to build systems that allow groups to realize these principles. The effectiveness of the systems is generally determined by the degree to which these principles are supported.

WinCuts [38] allows users to replicate a local window's pixel data to other devices (e.g. large shared displays). This allows sharing of task information, but does not allow for joint interaction. LiveMeeting [2] and Community Bar [39] allow users to share each other's desktop screens and interact with them. However, these systems do not support the ability to share applications from multiple devices at the same time. For example, this would not allow two users to view and compare each other's ideas in parallel.

CoWord [43] and similar frameworks [22] allow the same or similar application to be tightly synchronized through an underlying protocol. This allows for individual work in parallel as well as relaxed WYSIWIS [33]. However, because these protocols must be developed per-application, using this technique to support applications in MDEs would require an overly large effort. Another set of systems including KidPad [4], PointRight [20], Mighty Mouse [8], Dynamo [18], and Swordfish [17] allow joint interaction within a shared visual workspace, but do not allow individual work to be performed in parallel.

iRoom [19] allows information to be shared across devices by passing descriptors of content (e.g., a URL) between devices. Limitations of this approach include that it requires similar applications to be installed, it does not maintain applications' interaction context when relocated, and ownership of relocated applications is not maintained. This can inhibit the natural flow of group work. Colab [33] and i-LAND [36] support specific group activities within MDEs (e.g., shared note taking). However these systems do not support existing off-the-shelf applications and do not provide control over which applications can be shared.

Many usability studies of systems that support MDEs have been conducted (e.g., [6, 20, 25, 31]). These studies have typically focused on testing interface representations [6], relocation techniques [25], and coordination policies [24]. Other studies have compared the effectiveness of group work within MDEs to other workspace configurations [37].

Our work makes two contributions to this body of research. First, we present a new framework that allows principles of group work to be more fully realized within MDEs. For example, our framework supports joint interaction on both personal and shared displays, the ability to multitask among shared applications, and the ability to leverage any off-the-shelf application. A lightweight interface is also provided for choosing which applications can be shared and when.

Second, we report results from one of the first field studies investigating how groups leverage MDEs for *authentic* tasks. As a starting point, we studied how our framework is used for face-to-face software development. This allowed us to study in-depth how groups use and benefit from MDEs and how supporting frameworks can be improved.

Emerging Software Development Practices

Software developers are attempting to improve their work practices to meet the increasing demand for dependable systems [29]. A radical change is occurring in how the act of programming and related development activities is being performed. Development teams are rapidly transitioning from individuals working in their own offices to small groups working face-to-face in co-located workspaces.

As shown in Figure 1, these workspaces are typically configured with individual work areas but are also equipped with large displays, whiteboards, and other instruments to foster team communication and awareness. Early evidence suggests that the group activities that are facilitated by these workspaces can reduce defects in software and improve the quality of its overall design [21, 26, 32, 41, 42].

Though being situated within the same workspace allows for increased communication, it exacerbates the need to realize principles of group work. For example, groups need to share and interact with each other's task artifacts such as code editor windows, debug windows, and web browsers showing examples. Our work contributes a new framework that developers can use to more fully realize principles of group work within these types of co-located workspaces.

Tools for Supporting These Practices

Several tools have been designed or could potentially be adapted for face-to-face group development activities. For example, source code repositories like CVS [1] and SVN [3] can be used to help coordinate access to shared code. However, these systems typically embody formal processes, and do not provide an effective means for informally sharing task artifacts during group development activities.

A file server or e-mail can be used for sharing task artifacts. But, these approaches are not sufficient because they do not allow joint interaction or retain their interaction context when passed between users. Using personal devices, with one connected to a large display, is also insufficient, e.g., artifacts from multiple users cannot be shown in parallel.

Tools have been created for better coordinating activities among programmers. For example, Palanír [28] and Augur [14] provide visualizations of recent actions within a shared code repository. FASTDash [7] extends this awareness to include developers' actions within their local integrated development environments (IDEs). Collaborative IDEs, such as Jazz [9], allow users to see who is working within the shared code, receive updates of their actions, and chat with each other. Our framework can be used to complement many of these tools. For example, FASTDash could be launched from a personal device and placed on a shared display for maintaining awareness of group activity.

IMPROMPTU

IMPROMPTU (IMPROving MDE's Potential to support Tasks that are genUine) is a new interaction framework that allows users to better realize natural and effective group work practices when working in MDEs. Important benefits include the ability to share any off-the-shelf application without modification, to share applications and input across multiple displays, and to allow different users to place applications on large displays at the same time. It also affords a lightweight interface for performing these actions.

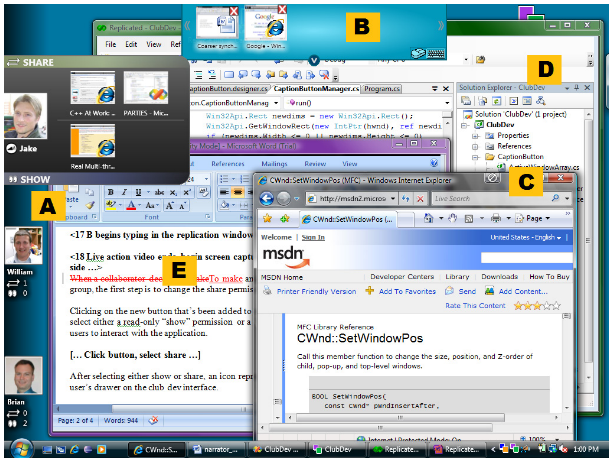


Figure 2: Screenshot of the IMPROMPTU user interface, along with replicated and local application windows on a user's machine. The collaborator bar is on the left (A), and one collaborator drawer is expanded showing the applications available to the group. The shared screen dock (B) allows windows to be placed on a large shared display. Whether an application is available to the group and what level of control is allowed can be set using (C). A replicated window in share mode allows interaction with its content (D); while a replicated window in show mode allows a user to view, but not modify its content (E).

While existing systems allow subsets of these benefits to be realized [2, 19, 33, 36, 38], a contribution of our framework is that it enables all of them to be more fully realized.

The framework's design was based on lessons learned from a series of surveys, interviews, and low fidelity prototyping sessions with professional developers. Our design also leveraged lessons from our earlier experiences developing and evaluating several interfaces for MDEs [5, 6] as well as principles from group work theory [12, 23, 30, 35].

User Interface

The interface provides a visual representation of group members, available large displays, and applications that have been made available to the group. See Figure 2. The user interface is comprised of the Collaboration Control, Collaborator Bar, and Shared Screen Dock(s).

Collaboration Control. This control allows a user to configure whether an application window is available to the group, and if group members are allowed to modify or only view its content. See Figure 3. The control is displayed on the title bar of every top-level application window. This location reinforces that this is a window-level operation, provides quick access to the functionality, and provides a persistent indicator of the window's sharing state.

Selecting the control reveals three sharing states (Fig. 3):

- *Do not show or share.* The application window is not available to group members (this is the default value).
- *Show.* The window is available to the group, but in a view-only mode. A live thumbnail of the window is

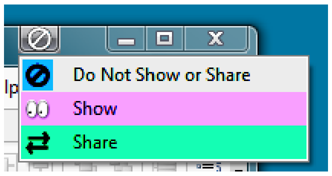


Figure 3: The collaboration control is displayed on top-level application windows. It is used to configure whether the window is available to the group, and whether group members can only view (Show) or interact with it (Share).

displayed in the *show* area of the user's representation in each group member's Collaborator Bar.

- *Share*. The window is available to the group and anyone can interact with its content. A thumbnail of the window is displayed in the *share* area of the user's representation in each group members' Collaborator Bar.

Offering both *show* and *share* is necessary as we have found that users have a strong sense of ownership of their applications. For example, a user can set a window to *show* to allow others to view and maintain awareness of their activity in relation to that window, but not be able to interact with it. Alternatively, with *share*, group members can edit source code or other documents together, or a user could pass control temporarily to another group member.

Collaborator Bar. This interface component provides a representation of each user participating in the collaborative session and the application windows that each user has made available to the group. When a user joins a session, their photo (or other image) appears within the Collaborator Bar, located on the side of the screen. See Figure 2 (A).

Each user's representation in the Collaborator Bar has a drawer with two rows. The top row displays thumbnails of application windows that have been set to *share* while the bottom row displays thumbnails of windows that have been set to *show*. Moving the cursor over a user's image causes the corresponding drawer to animate out. See Figure 4.

From a group member's drawer, a user can drag the desired thumbnail and drop it onto the desktop, establishing a *replication* of that window. For example, in Figure 2, a developer has replicated a team member's Visual Studio window. If the owner sets the window to *share*, the user

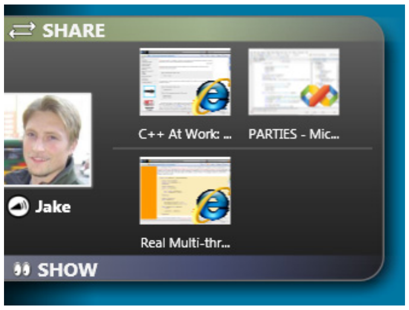
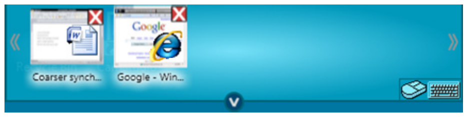
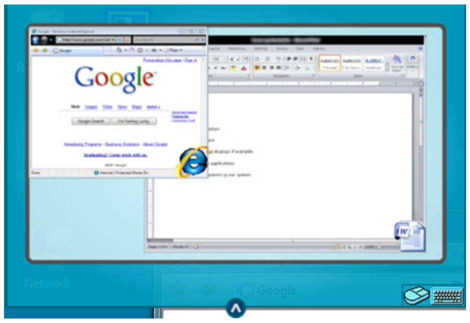


Figure 4: A close-up of an expanded collaborator drawer. The user has two applications in *share* mode (Internet Explorer and Visual Studio) and one in *show* mode (IE).



(a)



(b)

Figure 5: The shared screen dock. Hovering over it gives a summary of windows placed on the corresponding shared display (a). Selecting the arrow causes it to expand, providing a view that allows windows to be repositioned on the shared display (b). Selecting the redirection button redirects mouse/keyboard actions to the shared display.

could edit the source code while the owner switches to another task, or the two users could edit the code together. Users can also see each other's cursors within replicated windows. This particular cue establishes presence, provides awareness, and improves coordination [16].

One interesting aspect of this interaction is that it embodies a natural negotiation process. For example, it is the owner of an application window who determines if it is available to the group, while it is each group member who decides if and when to create a replication of that window.

Shared Screen Dock. This enables users to place application windows on a shared display, organize windows remotely, and redirect input to interact with them. Our system can support any number of shared displays, and each display is represented by its own dock. Any user can place any number of replicated windows onto the large displays.

The dock is minimized by default, and opens when the user moves the cursor over it. When opened, the dock shows thumbnails of all application windows on the corresponding display. In this view, the thumbnails are shown left-to-right, as this allows all of them to be seen at once without occlusion (Figure 5a). When expanded (via the button at the bottom), the dock shows a world-in-miniature representation of the content on the corresponding display (Figure 5b). By interacting with the thumbnails in this view, any user can adjust the position or z-order of the windows.

Any application window that has been set to *show* or *share* can be placed on a large display. From a group member's representation in the Collaborator Bar (including her own), a user drags the representation of the desired window and drops it onto the appropriate shared screen dock. The dock expands and the user can position the window as desired. It is important to note that a shared display can contain replicated windows from different users *at the same time*.

Users can also redirect their local input to a shared display. For example, this would allow group members to collectively interact with a replicated window *and* share the same visual focus. To redirect input, the user selects the redirection button on the screen dock. Input is returned to the local device using the *ctrl+alt+home* key sequence.

Interestingly, our interface centers on a view of the people participating in the collaboration rather than a strict spatial representation of the workspace, as done in much prior work (e.g. [5, 31, 40]). This was one of the primary lessons learned from working with users in our design process.

Implementation

A central goal of our implementation was to allow sharing of and interaction with any off-the-shelf application across devices. This is important because it would allow users to continue using the applications that they prefer and need for their daily work. For example, in our target domain of software development, users stressed that they use a large number of diverse applications and that building only application specific solutions would be of limited value.

To meet this goal, we chose a replication-based model for sharing applications. At a high level, this model works by capturing application windows' pixel data and reproducing it on other devices. This approach has some limitations, e.g., users cannot independently manipulate the view within a shared application, but the ability to utilize any off-the-shelf application in an MDE is a worthwhile tradeoff.

As a user interacts with a replicated view, the local input is captured and routed to the corresponding UI control in the source window. This provides a modest technical improvement over existing redirection techniques (e.g., [2]), but it allows substantial improvement in usability. For example, our technique allows group members to interact with a shared application while the owner is still able to interact with other applications simultaneously.

Our framework is implemented in managed C#, with some of the lower-level components in unmanaged C++. It has been tested with Windows Vista and XP and the techniques could be mapped onto other commonly used systems.

FIELD STUDY

We conducted a field study to evaluate our framework. A field study was conducted because this would allow us to better understand how groups utilize our framework for their *own* collaborative activities in *authentic* settings, how collaborative behaviors are impacted, and how such frameworks might be improved. All of this would have been very difficult to achieve in a single session lab study.

Study Participants

We recruited development teams from Microsoft Corp., a U.S.-based software company. To recruit teams, we sent a study solicitation to a company mailing list. We followed up with teams that expressed interest by arranging short meetings to outline for them the goals of our study, discuss requirements for participation, and describe the data that would be collected. These meetings also allowed the researchers to establish an initial level of trust with the teams. We found this to be especially important to ensure that our presence observing their daily activities for several weeks would not be overly disruptive or uncomfortable.

Through this process we identified two teams that met our requirements and agreed to the conditions of the study:

Team Alpha is a group of 9 individuals (all male), including 3 developers, 3 testers, a technical writer, a program manager, and a software architect. The team works closely together to create sample applications and documentation that promote the use of Microsoft development technologies. When we observed them, they were actively working on a package to showcase Web 2.0 technologies.

All members of *Team Alpha* were physically co-located within the same workspace (see Figure 1). The space was also equipped with a large display to which all members had access. This configuration supported the team's heavy adoption of Agile practices [29] which promote activities such as paired programming, daily status meetings (a.k.a. "scrums"), and face-to-face communication.

Our second team, *Team Beta,* is a group of 6 individuals (2 female) including 3 developers, 2 testers and a program manager. *Team Beta* is a feature team that works on next generation software management tools. We observed this team in the late stage of their development process as they were performing final testing, bug fixes, and integration.

Unlike the other team, *Team Beta* had individual offices located within close proximity (just a few steps away). Team members worked in their own offices, but frequently traveled to each other's offices to collaborate, as well as to a group conference room that was equipped with a large display. Like *Team Alpha*, this team performed many tasks collaboratively, including editing, debugging, and review.

Each team member was provided with a software gratuity for participating in the study. Also, to provide incentive for filling out questionnaires (discussed in the next section), each submitted questionnaire served as an entry into a raffle for two Zune™ MP3 players, one for each team.

Procedure and Measures

The study was conducted over a three week period and used a split pre/post observation design. We observed the teams without IMPROMPTU for one week to gain a baseline of their current collaborative practices. The next two weeks were spent observing the teams using IMPROMPTU.

Observations were split into alternating half-day sessions. That is, *Team Alpha* was observed in the morning and *Team Beta* in the afternoon. The next day, *Beta* was observed in the morning and *Alpha* in the afternoon, and so forth.

Category	Classification
Activity	Cognitive Synchronization
	Evaluation
	Explore Alternative Solutions
	Conflict Resolution
	Management
Domain	Edit
	Debug/Test
	Review
	Reference
	Design
Physical Movement	No Movement
	Move to Personal Device
	Move to Large Display
	Other
Use of Devices	Single Personal Device
	Multiple Personal Devices
	Personal Device(s) and Large Display
	Large Display Only
Time	Length of collaborative collaboration
Size	Number of individuals in collaboration

Table 1: Category and classifications used for coding observed instances of collaborative engagements.

For each session, at least one observer was present in the workspace to code the teams' collaborative activities. For approximately half of the sessions, there was an additional independent observer who also coded the activities, which allowed for cross-validation. Two independent observers were used during the course of the study. Each had over 20 years experience in behavioral data collection.

Observation Data

The observational coding scheme consisted of 6 primary categories, summarized in Table 1. The categories were based on schemes used in previous studies of collaborative software activities [7, 11]. After several iterations, we arrived at a scheme that we felt provided an adequate (for our purposes) representation of the teams' collaborative behavior, was simple and quick enough to use in real-time, and allowed the data to be easily quantified and compared.

Instances of collaborative engagements that could be externally observed were coded. An instance would be signaled by initiation of physical movement to each other's work areas and/or verbal communication. Instances were considered complete when the verbal communication ended and/or users returned to their own work areas.

A brief example will illustrate how the scheme was applied. Suppose one user physically moves to another's device and they work together to solve an error in the code. Along with modifying the code, they consult discussion forums and API documents to explore solutions. This instance would be coded as Evaluation and Explore Alternative Solutions under Activity; Edit and Debug/Test under Domain; Move to Personal Device under Physical Movement; and Single Personal Device under Use of Devices.

Inter-coder reliability was measured by sampling 10% of the coded data and computing Cohen's κ [10], a common measure of coder reliability. All but two categories (Review and Edit under Domain), had Cohen's $\kappa \geq 0.76$, which indicates a very high degree of agreement. The remaining categories had lower Cohen's κ (0.25 & 0.46, respectively), but this was due in part to having unbalanced categorization frequencies. A further test of raw agreement showed that these categories had > 80% consistency, and we believed this represented sufficient agreement.

Instrumented Data

We instrumented our framework to log usage data. This was done to understand which features were used and how often. This data included which applications had been shared or shown and for how long, which applications had been replicated, to which device they were replicated and for how long, how often input redirection was used, and how often users interacted with the Collaborator Bar.

User Feedback

Finally, we collected user feedback in two forms. First, we asked users to complete a questionnaire that probed their use of the framework for a recent, meaningful collaborative activity. Questions included explaining the motivation for the activity, who was involved and their role, which aspects of the framework were used, and how it affected the overall activity. Each team member was asked to complete the questionnaire every other day using an online form.

In addition to the questionnaires, we performed a 30 minute group interview with each team at the end of the study. We asked the teams about their overall experiences using the framework, what they felt were its strengths/weaknesses, and for recommendations on how it could be improved.

RESULTS

In this section, we describe how the teams utilized our framework, provide results from the instrumentation data, and discuss impacts on existing collaborative practices.

Use of IMPROMPTU

During the study, our framework was leveraged to perform myriad collaborative tasks. These tasks included providing opportunistic assistance in solving complex complier errors, working closely together to integrate different users' code into a single solution, reviewing API documentation, and brainstorming designs for new features. In support of these tasks, users leveraged our framework to show or share a wide variety of applications, including source code and document editors, communication tools, and Web browsers. The applications were replicated between personal devices as well as placed on the large displays. A few detailed examples will help exemplify the use of the framework.

In one instance, we observed John, a developer from *Team Alpha,* encounter a compiler error while working on his code. After spending a few minutes attempting to solve the error, he requested assistance from Stan, another developer on his team. John used the Collaboration Control to set his source code editor to *share* and Stan replicated the window by dragging its thumbnail from John's area within the Collaborator Bar. Stan interacted with John's code editor to review and understand the problem, used the telepointer to highlight segments of possible problem code for John, and offered verbal suggestions on how to proceed.

In another instance, the manager on *Team Beta,* Felix*,* needed to send a status report to his boss. Because he was unsure on technical details, he asked Susan, a developer on the team, to assist. Felix shared his document window, and Susan replicated it on her device. A verbal communication channel between the two was established via the office phone. Felix and Susan were able to jointly compose the report, each providing content that they knew most about.

In a similarly structured task, Steven and Greg, two developers on *Team Beta*, used the framework to review and integrate Steven's code into the source repository. Greg replicated a code editor and a separate integration tool that Steven had *shared*. Greg spent a few minutes reviewing Steven's code, while Steven multitasked between answering an occasional question from Greg and his email. When Greg finished reviewing a section of code, the two rejoined working in the code editor, made modifications to the code, and then moved on to the next section.

Aside from peer-to-peer tasks, the teams also engaged in group-wide tasks. Members of *Team Alpha*, for example, conducted a team-wide design session. In the session, the lead developer replicated an architectural diagram from his personal device onto the group's large display. Using the diagram as a shared visual reference, the team stood near the large display to discuss the content of the diagram and its implications on their immediate and future work.

In another team-wide collaboration that was focused on feature planning, a developer replicated a note taking application onto the large display and the team took turns adding content to the list. Team members added content by interacting with replications of the note taking application that they had also created on their personal devices.

Finally, one of the more inventive uses of the framework that we observed was to place information on the large display to *passively* attract the attention of team members. For example, in *Team Alpha*, a developer placed a trade news article discussing a competing product on the display to entice discussion and comments from team members.

Though not exhaustive, these examples demonstrate that teams did indeed utilize the framework to perform many different types of collaborative tasks. Users were able to quickly and easily make task-related artifacts available to the group, without disrupting the natural flow or pace of the collaboration. For example, users could, in most cases, quickly transition from individual work to joint interaction with an application without having to physically move or reconfigure devices. For example, as one user commented, he appreciated "the ability to have multiple team members actively manipulate the same application."

Users also appreciated the ability to share windows while maintaining the ability to multitask with other applications. As one user wrote, "I like that I can be actually working on another team member's machine (reviewing code, editing a PowerPoint, editing a word doc) in a hidden window (off to the side) while not blocking them from doing other work."

Users could also quickly make information available for group review and comparison through the ability to place applications from multiple devices onto large displays. One user said the most useful piece of functionality was "the ability to easily display items [from] the desktop to [the large display] where they could be viewed by others." Another said it was useful because "you could see that someone was collaborating, and might choose to jump in."

Instrumented Measurements

We collected over 1035 hours of system use across the 15 users. The amount of use by each user was highly varied; the overall mean length was just under 74 hours (SD=64h04m). The distribution, though, was bimodal; a subgroup of 7 users used the framework for a mean of 16h21m (SD=3h46m) during the course of the study, while the remaining 8 had a mean of 131h38m (SD=17h24m). These results exemplify two types of use of the framework. The less frequent users would tend to only launch the framework when it was needed (e.g., to initiate or support a replication) while others would launch and leave it running throughout the entire workday (e.g., to make applications available to facilitate awareness for team members).

96 applications were made available to the group in total; 74 were *shared* and 22 were *shown*. On a per user level, mean *shares* was 6.73 (SD=10.1) and mean *shows* was 2.0 (SD=2.1) for the period the framework was deployed. This result illustrates that users typically only made applications available to the group when there was an immediate need. It also indicates that the owner of an application deliberately considered the control other members should have over it.

Figure 6 shows the distribution of application types made available. Given the task domain, it was expected that the majority of applications made available would be source code related. But, many other applications were also made available including instant messaging (IM) applications, diagramming tools, and Web browsers. This highlights the need for MDE frameworks to support myriad applications.

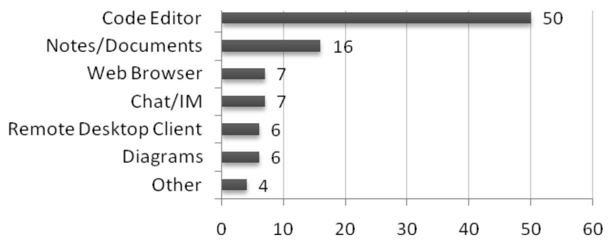


Figure 6: Breakdown of the type of applications that users made available to their group.

Of the 96 (~80%) applications that were made available, 77 of them were replicated. The mean time that an application was replicated was 13m38s (SD=23m33s). 19 (~25%) of the replications occurred in collaborations in which more than one application was replicated at the same time. This functionality was leveraged, for example, to compare information that was distributed across several devices.

At the device level, the median number of applications that were replicated *from* and *to* each device was 6.00 (SD=4.0) and 2.50 (SD=9.7), respectively. The imbalance is due to the heavy use of the large displays (35/77, or 45.5% of the replications). This shows that both personal-to-personal and personal-to-large display replications were utilized.

There were 898 Collaborator Bar interactions, a per user mean of 68.9 (SD=41.9). This shows users were leveraging this component as a means for establishing replications and for acquiring awareness of others' current tasks.

Finally, the users rarely used the input redirection support (only 3 instances logged). This result indicates that users strongly preferred to perform input actions via replications on their local device, rather than on the shared display.

Impact on Existing Collaborative Practices

We collected 125 hours of coded observational data over the three week period of the study. 50 hours of observation were conducted prior to the introduction of our framework, and 75 hours were conducted with our framework. The coded data is summarized in Figure 7.

For Physical Movement, a change was found in the number of movements to a single personal device in *Team Beta*. Movements dropped from 66% to 40%. No differences were found for *Team Alpha*. Analysis did not reveal significant differences in the other categories.

We do not believe this is a negative result for two reasons. First, the framework was specifically designed to support existing collaborative practices, not to necessarily change them. Second, the use of the framework was studied within teams who had well-established working relationships (e.g. the teams had been working together for several years). The fact that the teams used the framework in support of many tasks without a detectable change in observed behavior suggests that the design of the framework succeeded in supporting natural collaborative practices. Indeed, as we found in the interviews, users stated that a central benefit of the framework was that it allowed new opportunities for collaboration within their existing group practices.

This result does not necessarily mean that changes in group behavior would not occur. Rather, it suggests that such changes would occur gradually, over much longer duration. We plan to further investigate this issue in future studies.

DISCUSSION

A central design decision in developing our interaction framework was to use a replication model. The purported benefit of this model is that it would allow any off-the-shelf application to be shared across devices. Indeed, results from our study showed that this design decision was justified, as users chose to leverage this benefit to share code editors, Web browsers, and document editors, among others. Another benefit is that the model enables the interaction context of applications to be maintained when replicated (e.g., keeping breakpoints and stack data in a debug tool). This proved to be valuable, as it minimized the overhead when transitioning between individual and group work.

A main criticism of this approach is that users cannot independently manipulate the view of a shared application. However, results from our study did not show this to be a serious problem in the domain studied. The reason is that our framework was used mostly to support opportunistic, short-lived collaborative engagements. In these cases, it was important for users to maintain the *same* view.

Overall, our study showed that groups did find value in using our framework (in an MDE setting) to support software development. This is evidenced by the wide range of tasks performed and applications shared, and by many users stating during the group interview that they would want to continue using the framework. Several users also

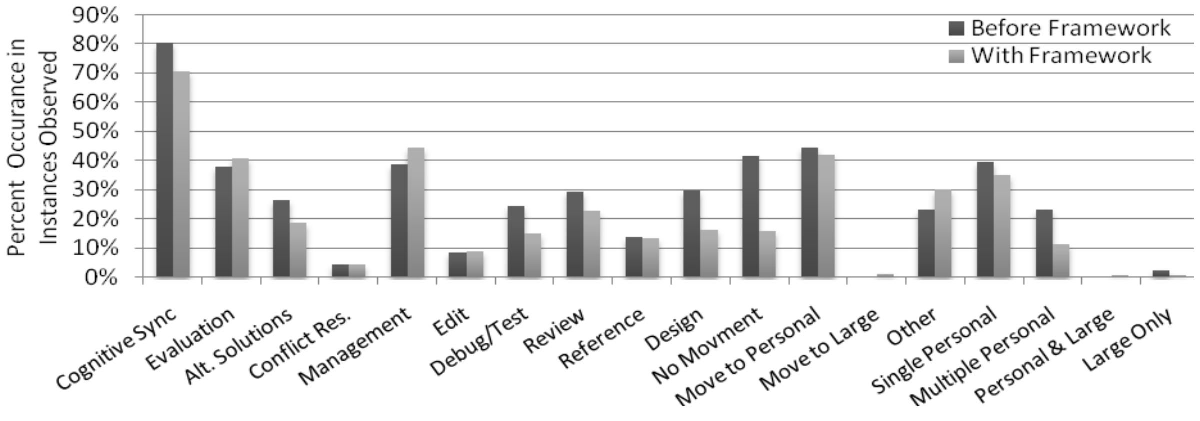


Figure 7: This chart summarizes the observation data gathered before and after the framework was deployed. Results are reported as the percentage of total collaborative instances observed per condition (194 before/160 with framework).

stated that our framework allowed the large displays to be more tightly integrated into their work practices, as their use could now be shared without configuration overhead.

Teams utilized almost all of the features of our framework, but its usage frequency was relatively modest overall. We attribute this usage behavior mostly to the task domain, as users were often focused on their own individual efforts. The value of the framework was thus derived not from its frequency of use, but in its utility in supporting specific instances of collaborative engagements. Users expressed, however, that the framework may have been utilized more frequently in situations where collective understanding of the technical content is low, e.g., during project planning or initial coding, resulting in more information being shared.

The use of our framework was studied in one complex task domain, software development. This decision was made because it would allow us to study the actual benefits and use of this type of framework in an authentic setting, filling a large gap in the current literature on MDEs. Groups in other domains that share similar work processes, e.g., to opportunistically share task information, transition between individual and joint work, and collectively manipulate content on large displays, may benefit from and/or use our framework in similar ways. Collaborative design and creative writing offer potential domains in which to further study the use of our (or a similar) framework.

Though studied in one domain, results from our study did reveal opportunities for improving the design of MDE frameworks in general. One improvement would be to enable the owner of a window to know the status of the associated replications on other users' machines. For example, users wanted to know when a team member was actively working within a replicated window or if it had been moved out of focus or minimized, as this would allow them to better time appropriate transitions to (sub)group work. One possible solution would be to provide visual cues of replication status within the title bar or other decoration of the source window on the owner's device.

A second improvement would be to merge the framework's Collaborator Bar with the contact lists already available in existing communication tools such as e-mail and chat. This would reduce the need for maintaining separate collaborator lists and free up valuable screen real estate. It could also enable cross-utilization of collaboration tools. For example, an ongoing group chat could easily expand, if desired, to a collaboration involving several application windows shared among the users' displays.

Finally, frameworks should enable users to manage sharing options from any device within the MDE. For example, once engaged in a task at another user's work area or large display, a user may want to access applications running on her personal device. The current design of our framework requires the user to move back to her device to set the appropriate permissions. A possible solution would be to allow the user to enter a password into her representation on the Collaborator Bar to set all running applications on her local device to be shared (or a subset based on defined rules), allowing the desired applications to be available.

CONCLUSION

Multiple display environments offer an exciting opportunity for allowing groups to collaborate more effectively with digital information. In this work, we have made several important contributions towards realizing this vision.

First, we have described the design and implementation of a new interaction framework that enables core principles of effective group work to be better realized in multiple display environments. Key benefits include the ability to quickly share task information via existing off-the-shelf applications, to replicate application windows and input across devices to support focused problem solving, and to place task information on shared displays for discussing and comparing ideas. The framework also affords a lightweight interface for performing these and other related actions.

Second, we have presented results from one of the first field studies investigating how groups utilize an interaction framework for MDEs in support of authentic collaborative tasks. We deployed our framework and, over a three week period, studied how it was utilized by two software development teams. Results show that teams leveraged the framework most for facilitating opportunistic collaboration and found value in using it overall. From observing how groups used the framework, we produced several insights into how the design of such frameworks can be improved.

Finally, we have made our framework publicly available. This will enable practitioners to realize the benefits of our framework for their own collaborative activities and enable researchers to test new interaction designs or study its use in other task domains. The framework can be downloaded from http://orchid.cs.uiuc.edu/projects/IMPROMPTU/.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under award no. IIS 06-43512.

REFERENCE

1. CVS - Concurrent Versions System. Retrieved January 3, 2008 from: http://www.nongnu.org/cvs/
2. Microsoft Office LiveMeeting. Retrieved January 3, 2008 from: http://office.microsoft.com/livemeeting/
3. Subversion Project. Retrieved January 3, 2008 from: http://subversion.tigris.org/
4. Benford, S., et.al. Designing Storytelling Technologies to Encourage Collaboration between Young Children. *Proc. CHI*, 2000, 556-563.
5. Biehl, J.T. and B.P. Bailey. ARIS: An Interface for Application Relocation in an Interactive Space. *Proc. Graphics Interface*, 2004, 107-116.
6. Biehl, J.T. and B.P. Bailey. Improving Interfaces for Managing Applications in Multiple-Device Environments. *Proc. Advanced Visual Interfaces (AVI)*, 2006, 35-42.

7. Biehl, J.T., M. Czerwinski, G. Smith and G.G. Robertson. FASTDash: A Visual Dashboard for Fostering Awareness in Software Teams. *Proc. CHI*, 2007, 1313-1322.

8. Booth, K.S., B.D. Fisher, C.J.R. Lin and R. Argue. The "Mighty Mouse" Multi-Screen Collaboration Tool. *Proc. UIST*, 2002, 209-212.

9. Cheng, L.-T., S. Hupfer, S. Ross and J. Patterson. Jazzing up Eclipse with Collaborative Tools. *Proc. OOPSLA Workshop on Eclipse Technology eXchange*, 2003, 45-49.

10. Cohen, J. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, *20* 37-46.

11. d'Astous, P., F. Détienne, P.N. Robillard and W. Visser. Types of Dialogs in Evaluation Meetings: An Analysis of Technical-Review Meetings in Software Development. *Proc. Conference on the Design of Cooperative Systems*, 1998, 25-33.

12. Douglas, T. *Groupwork Practice*. Tavistock Publications Limited, London, England, 1976.

13. Elwart-Keys, M., D. Halonen, M. Horton, R. Kass and P. Scott. User Interface Requirements for Face to Face Groupware. *Proc. CHI*, 1990, 295-301.

14. Froehlich, J. and P. Dourish. Unifying Artifacts and Activities in a Visual Tool for Distributed Software Development Teams. *Proc. ICSE*, 2004, 387-396.

15. Grudin, J. Computer-Supported Cooperative Work: History and Focus. *Computer* (May): 19-26.

16. Gutwin, C. and S. Greenberg. A Descriptive Framework of Workspace Awareness for Real-Time Groupware. *Journal of Computer-Supported Cooperative Work* (3-4): 411-446.

17. Ha, V., K. Inkpen, J. Wallace and R. Ziola. Swordfish: User Tailored Workspaces in Multi-Display Environments. *Extended Abstracts CHI*, 2006, 1487-1492.

18. Izadi, S., H. Brignull, T. Rodden, Y. Rogers and M. Underwood. Dynamo: A Public Interactive Surface Supporting the Cooperative Sharing and Exchange of Media. *Proc. UIST*, 2003, 159-168.

19. Johanson, B., A. Fox and T. Winograd. The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. *IEEE Pervasive Computing*, *1* (2): 67-74.

20. Johanson, B., G. Hutchins, T. Winograd and M. Stone. Pointright: Experience with Flexible Input Redirection in Interactive Workspaces. *Proc. UIST*, 2002, 227-234.

21. Layman, L., L. Williams and L. Cunningham. Exploring Extreme Programming in Context: An Industrial Case Study. *Proc. IEEE Agile Development Conference*, 2004, 32- 41.

22. Li, D. and J. Lu. A Lightweight Approach to Transparent Sharing of Familiar Single-User Editors. *Proc. CSCW*, 2006, 139-148.

23. Mandviwalla, M. and L. Olfman. What Do Groups Need? A Proposed Set of Generic Groupware Requirements. *ACM Transactions on Computer-Human Interaction*, *1* (3): 245-268.

24. Morris, M.R., K. Ryall, C. Shen, C. Forlines and F. Vernier. Beyond "Social Protocols": Multi-User Coordination Policies for Co-Located Groupware. *Proc. CSCW,* 2004, 262-265.

25. Nacenta, M.A., D. Aliakseyeu, S. Subramanian and C. Gutwin. A Comparison of Techniques for Multi-Display Reaching. *Proc. CHI*, 2005, 371-380.

26. Nosek, J.T. The Case for Collaborative Programming. *Communications of the ACM*, *41* (3): 105-108.

27. Rekimoto, J. and M. Saitoh. Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments. *Proc. CHI*, 1999, 378-385.

28. Sarma, A., Z. Noroozi and A.v.d. Hoek. Palantír: Raising Awareness among Configuration Management Workspaces. *Proc. ICSE*, 2003, 444-454.

29. Schwaber, K. and M. Beedle. *Agile Software Development with Scrum*. Prentice Hall, Upper Saddle River, NJ, 2002.

30. Scott, S.D., K.D. Grant and R.L. Mandryk. System Guidelines for Co-Located Collaborative Work on a Tabletop Display. *Proc. ECSCW,* 2003, 159-178.

31. Shen, C., K.M. Everitt and K. Ryall. Ubitable: Impromptu Face-to-Face Collaboration on Horizontal Interactive Surfaces. *Proc. UbiComp*, 2003, 281 - 288.

32. Souza, C., D. Redmiles and P. Dourish. "Breaking the Code", Moving between Private and Public Work in Collaborative Software Development. *Proc. CSCW*, 2003, 105-114.

33. Stefik, M., D.G. Bobrow, G. Foster, S. Lanning and D. Tatar. Wysiwis Revised: Early Experiences with Multiuser Interfaces. *ACM TOIS*, *5* (2): 147-167.

34. Stefik, M., G. Foster, D.G. Bobrow, K. Kahn, S. Lanning and L. Suchman. Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. *Communications of the ACM*, *30* (1): 32-47.

35. Steiner, I. *Group Process and Productivity*. Academic Press, New York, 1972.

36. Streitz, N.A., J. Giessler, T. Holmer, S. Konomi, C. Muller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz and R. Steinmetz. I-Land: An Interactive Landscape for Creativity and Innovation. *Proc. CHI*, 1999, 120-127.

37. Streitz, N.A., P. Rexroth and T. Holmer. Does Roomware Matter? Investigating the Role of Personal and Public Information Devices and Their Combination in Meeting Room Collaboration. *Proc. ECSCW*, 1997, 297-312.

38. Tan, D.S., B. Meyers and M. Czerwinski. Wincuts: Manipulating Arbitrary Window Regions for More Effective Use of Screen Space. *Proc. CHI*, 2004, 1525-1528.

39. Tee, K., S. Greenberg and C. Gutwin. Providing Artifact Awareness to a Distributed Group through Screen Sharing. *Proc. CSCW*, 2006, 99-108.

40. Wigdor, D., C. Shen, C. Forlines and R. Balakrishnan. Table-Centric Interactive Spaces for Real-Time Collaboration. *Proc. Advanced Visual Interfaces (AVI)*, 2006, 103-107.

41. Williams, L., R. Kessler, W. Cunningham and R. Jeffries. Strengthening the Case for Pair Programming. *IEEE Software*, *17* (4): 19-25.

42. Wu, J., T.C.N. Graham and P.W. Smith. A Study of Collaboration in Software Design. *Proc. International Symposium on Empirical Software Engineering*, 2003.

43. Xia, S., D. Sun, C. Sun, D. Chen and H. Shen. Leveraging Single-User Applications for Multi-User Collaboration: The CoWord Approach. *Proc. CSCW*, 2004, 162-171.