

Flexible User Interfaces for Group Collaboration

Ivan Marsic

Bogdan Dorohonceanu

Center for Advanced Information Processing (CAIP)
Rutgers—The State University of New Jersey

Flexible user interfaces that can be customized to meet the needs of the task at hand are particularly important for telecollaboration. This article presents the design and implementation of a user interface for DISCIPLINE, a platform-independent telecollaboration framework. DISCIPLINE supports sharing of Java components that are imported into the shared workspace at run-time and can be interconnected into more complex components. As a result, run-time interconnection of various components allows user tailoring of the human-computer interface. Software architecture for customization of both a group-level and application-level interfaces is presented, with interface components that are loadable on demand. The architecture integrates the sensory modalities of speech, sight, and touch. Instead of imposing one "right" solution onto users, the framework lets users tailor the user interface that best suits their needs. Finally, laboratory experience with DISCIPLINE tested on a variety of applications with the framework is discussed along with future research directions.

1. INTRODUCTION

Designing flexible and dynamically configurable user interfaces (UIs) is challenging, and it is unlikely that developers will come up with a solution for all problems that is appropriate for all users. Users often like to customize their interfaces, as evident from different preferences in office and desktop designs. The desktop paradigm for productivity applications (spreadsheets, word processors, etc.) offers one user interface for all users. Although this may be tolerated in an office environment, newly emerging mobile environments require diverse user interfaces (Smailagic & Siewiorek, 1996). This article focuses on flexible UIs for synchronous

Research contributors to this project include Professors J. Flanagan and A. Medl as well as B. Sletterink. The research reported here is supported by NSF STIMULATE Contract No. IRI-96-18854, DARPA Contract No. N66001-96-C-8510, NSF KDI Contract No. IIS-98-72995, and the Rutgers Center for Advanced Information Processing (CAIP).

Requests for reprints should be sent to Ivan Marsic, Center for Advanced Information Processing (CAIP), Rutgers—The State University of New Jersey, 96 Frelinghuysen Road, Piscataway, NJ 08854-8088. E-mail: marsic@caip.rutgers.edu

telecollaboration that can be dynamically adapted to the user's needs. Our approach offers architecture for end-user customization through on-demand loadable software components and XML documents. The architecture is an integral part of DISCIPLÉ (Distributed System for Collaborative Information Processing and Learning; Marsic, 1999; Marsic & Dorohonceanu, 1999). DISCIPLÉ provides a different look and feel not only across different groups but also across individuals within a group.

Tailorable interfaces are abundant. However, most of them focus on low-level interface choices such as fonts, colors, display resolution, cursor shape, mouse speed, mouse button choice, gamepad button choice, and function key mapping. The tailorable interface presented here has functionality that the user can exploit to specify his or her preferences at more abstract levels to customize data presentation methods, interaction preferences, and so forth.

A groupware infrastructure is defined by three dimensions (Usability 1st, 1999): communication (pushing or pulling information out into an organization), collaboration (using shared information and building shared understanding), and coordination (concurrency control, latecomer support). The user interface for groupware requires understanding of groups and how people work in teams and understanding of networking technology and how aspects of that technology (e.g., delays in synchronizing the views) affect a user's experience. The solution presented here offers flexibility both at the group level and at the application level. At the group level, the shared workspace can be adapted by loading collaboration-specific Java components that incorporate the collaboration and coordination dimensions into the user interface. At the application level, the user can choose between multiple sensory modalities to interact with the application.

Although human-computer interfaces have significantly advanced over the past decade, most people still only use display, mouse, and keyboard. Alternative modalities for human-computer communication, such as speech, force-feedback tactile interfaces, and gaze tracking, have proven to be convenient and efficient to use for specific tasks (Cohen et al., 1996; Flanagan & Marsic, 1997). By harnessing the potential power of these modalities, application developers may be able to increase the efficiency and accessibility of applications for people who do not use computers daily as well as for people who are disabled. Hands-free interfaces are also of interest for applications in environments such as knowledge-based systems for on-site technicians, emergency service personnel, and soldiers in the field, which do not allow the use of conventional modalities.

There is a natural match between the type of applications suitable for telecollaboration and those suitable for multimodal interaction. Telecollaboration applications require significant spatial and graphics content manipulation; otherwise, telephone conversations and e-mails would suffice. A multimodal interface that includes gestures and manipulation using modalities other than keyboard and speech is naturally suitable for this type of application. Thus, it is natural to investigate telecollaboration in conjunction with multiple modalities.

This article is organized as follows. First, the state of knowledge and work in progress in the area of flexible group communication multimodal interfaces are reviewed. Interface flexibility at the group interaction level then is addressed. The next section focuses on providing flexibility to interact with applications via a

multimodal human–computer interface. The following section discusses dynamic interface customization. Finally, test applications and experimental results as well as future research directions are presented.

2. RELATED WORK

Early efforts in tailorable groupware applications and interfaces were presented by Bentley, Rodden, Sawyer, and Sommerville (1992) and Malone, Lai, and Fry (1992). Syri (1997) proposed a concept for the development of generic groupware with the possibility to configure its basic cooperation support functionality. Because DISCIPLÉ is based on JavaBeans, it supports tailoring of different cooperation functionalities, and this article in particular focuses on tailoring the interface functionality.

The model of collaboration used in DISCIPLÉ has certain similarities to the *locale* concept and its Orbit implementation (Mansfield et al., 1999). Orbit focuses on visualizing the content of locales, with some information about the states of shared artifacts, but it is not concerned with interface customization or the viewers and editors who interact with the artifacts. Unlike Orbit, DISCIPLÉ's primary goal is group sharing of interactive applications. In particular, it focuses on JavaBeans components (Sun Microsystems, Inc., 1999a) and allows interaction with the components, distributes their events remotely, and enables dynamic customization of user interfaces.

GroupKit toolkit (Roseman & Greenberg, 1996) is used in groupware prototyping for experimenting with different types of group awareness and modeling shared applications (called conferences). Its widgets are particularly relevant for relaxed WYSIWIS (What You See Is What I See) groupware awareness features: identity, location, and actions (Stefik, Bobrow, Lanning, & Tatar, 1987). The widget set comprises telepointers, radar views, location and action viewers, and multiuser scrollbars. As a toolkit, the system can be used to develop new applications, but it does not provide for run-time interface customization.

Flexible JAMM (Begole, Rosson, & Shaffer, 1999) supports sharing of single-user Java applets in synchronous collaboration. It relies on a custom-modified version of the Java Development Kit (JDK 1.2), which makes it nonportable. JAMM primarily targets unanticipated sharing or spontaneous collaboration where a user is able to initiate sharing at any time during the execution of an application, not only before an application is started. Due to its different collaboration model, JAMM does not provide visualization of the collaboration space (people and shared resources). The workspace does not accept more than one applet at a time or provide for user interface customization as the framework presented here does.

Microsoft NetMeeting (Microsoft Corporation, 1999a) allows sharing of an application's screen image by sending it to the other parties. Whereas DISCIPLÉ offers a hierarchical view of the collaboration space, NetMeeting has a very primitive concept of a collaboration space, through a directory server of on-line users. It does not offer support for either multiple sessions (there is only one session) or multimodal user interface management.

The TANGO system (Beca et al., 1997) is a framework that extends the capabilities of Web browsers toward an interactive, multimedia collaborative environment. It does not provide support for customizable interfaces and lacks a multimodal user interface.

Several multimodal applications and frameworks have been developed by other groups, but none of them support multiple concurrent applications. The examples of existing multimodal systems given subsequently are focused on developing an optimal multimodal interface for a specific task. Unlike this, DISCIPLÉ aims to develop a general architecture suitable for a wide variety of tasks that allows easy interface customization by end users and supports multiple concurrent applications (Marsic, Medl, & Flanagan, 2000).

A well-known multimodal groupware application is QuickSet (Cohen et al., 1996), which was built entirely on the concept of communicating agents and has pen and speech integration with a collaborative electronic map application.

VIENA Classroom (Winiwarter, 1996) is a multimodal collaborative distance learning application, where students can mark certain points in course material that they do not understand and ask a question in Japanese. An agent that searches a list of frequently asked questions, or the teacher, can then answer the question.

Gellersen (1995) proposed the MAUI library for facilitating multimodal interface development. It is not application specific and can be used for multiple applications, but it does not support multiple concurrent applications in a shared environment.

Quickset and VIENA Classroom are both applications with a single grammar and vocabulary, whereas the grammar and vocabulary of DISCIPLÉ's multimodal interface support multiple concurrent applications. Along with MAUI, none of these systems provide architectures for run-time customization of the user interface.

A key principle in DISCIPLÉ design is to allow flexible composition. The user can visually compose the shared applications as well as the user interface. Although some components of the framework presented here exist in other systems, the run-time composition of group awareness and multimodal interface components are unique to the DISCIPLÉ framework.

3. GROUP-LEVEL INTERFACE FOR COLLABORATION

A major problem in widespread acceptance of groupware is the difficulty of finding the collaborators and, once they are known, establishing a meeting session. User interfaces for creating sessions tend to be unfriendly. Groupware is linked with the needs of cross-organizational, time-driven, task-oriented, small, cohesive groups. When these groups share data, they have different representation metaphors.

Synchronous groupware comprises two main tasks: (a) session establishment and monitoring and (b) application and data sharing. To help users collaborate, the graphic-user interface (GUI) of the DISCIPLÉ framework consists of a desktop (the communication center that includes a view of the people and their relationships) and workspaces (views of meeting places and resources; Dorohonceanu & Marsic, 1999).

3.1. Collaboration Desktop

A key function of the user interface is effective visualization of the collaboration space. The collaboration space is essentially a “phone book” of all people with whom a user can collaborate. It is structured to reflect real-life relationships between professionals. It may be visualized in different ways, for example, by using an abstract model or a virtual model of the physical world. The traditional phone book only allows direct access to persons via an alphabetical listing, which has been found to be inadequate in many situations. Now that the world is becoming miniature, global, and mobile, the organization and representation of this contact structure need to incorporate many of these multiple representations but still handle the scalability problems of being global.

The simplest abstract representation shows a simple list of collaborative places. A more complex representation structures the places for collaboration into a tree or a graph, where the nodes at higher hierarchical layers correspond to buildings, meeting rooms, and so forth. An even more complex representation positions similar places proximally according to a certain distance measure. On the other hand, in a physical representation, the space is represented as a three-dimensional (3-D) virtual world, where the user walks through streets and hallways to reach a collaboration place. The places in DISCIPLÉ’s collaboration model are characterized by the topic of collaboration rather than by their physical location, and, therefore, an abstract representation is more appropriate than a physical one.

The desktop provides for workspace management functions, such as creating or entering a workspace and getting information about the existing workspaces and the active users inside them. It is the visualization medium for all the database information that is available about the users and their current activities. This medium has to be customized for specific application domains (i.e., to visualize the same abstract model under different views). Suppose a group of users want to use DISCIPLÉ in the military domain; then the desktop has to render the information in a military-specific format. Another group wants to use it in the medical domain, so the information has to be presented accordingly.

The dynamic state of any application that features a GUI has two representations: the view (on-screen representation) and the model (data representation; Krasner & Pope, 1988). In collaborative applications, the view, the model, or both can be shared. Most collaborative applications make no distinction between the view and the model. This becomes a problem when users have different interfaces (e.g., two-dimensional [2-D] vs. 3-D user interface, palmtop vs. workstation screen) or use different filters and metaphors to render the data. In the DISCIPLÉ framework, the model represents the database of the collaborating users, their membership in different groups, and shared applications. The views render the model data as domain-specific visualizations.

To support the notion of separation between the model and the different views when rendering the information about the collaboration space, the eXtensible Markup Language (XML) was used, which can handle different rendering styles (W3C Architecture Domain, 1999). The model is represented as an XML document, and the different views are specified by eXtensible Style Language (XSL) docu-

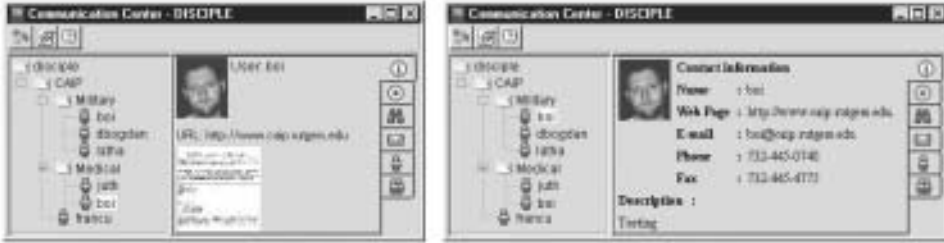


FIGURE 1 DISCIPLÉ communication center (desktop). The view of the collaboration space is structured into organizations, meeting places, and users. The user information is filtered and rendered by using different viewing styles; see, for example, the user *boi*, who entered the meeting places Medical (left) and Military (right). In the right window of the desktop, several utilities available to the user are tabbed: Besides the information about the currently selected item in hierarchy, common tools (such as chat window, video or audio conferencing startup, e-mail, and query window for finding users) are provided.

ments, provided for each group at creation time. At present, the collaboration space is rendered as a directory tree and only the place and user descriptions can be rendered in domain-specific ways. Figure 1 presents two different views of the profile for the same collaborating user (*boi*) who entered two meeting places (“Military” and “Medical”). Each meeting place reflects a specific view depending on the domain the users collaborate in, and information is filtered accordingly. Under Medical, only the user name, the address, and a snapshot of his personal Web page are shown, whereas under Military, full information about the user is provided.

3.2. Collaboration Workspace

The design of the collaboration workspace was started with the following observations (Usability 1st, 1999): First, organizing and scheduling groups are more difficult than organizing and scheduling individuals. Group interaction style is hard to characterize beforehand, whereas individual characteristics are often possible to determine before a study is conducted. Preestablished groups vary in interaction style, and how long they have been a group affects their communication patterns. New groups change quickly during the group formation process. Therefore, designing an interface that meets all users’ requirements is complicated by the facts that groupware users have different backgrounds and roles, the dynamics of collaboration are difficult to study and understand, and group behaviors cannot be generalized from one group to another. A groupware system should maximize human interaction while minimizing technology interference. Second, *session-room* and *building-floor-room* (MITRE Corporation, 1999; Roseman & Greenberg, 1996) are two well-known metaphors for collaborative space visualization. Opening a session and entering a room might be not so easy for a novice user. Corporate culture is changed by groupware technology that supports the team structure. A

real floor in a building is in contrast to the trend of future working places. Employees will work in groups, and even if they are at work, in an organization, in a real building, asking them to imagine themselves in another virtual building with virtual floors and virtual rooms could be a bit confusing.

Drawing on these observations, the authors provided components for users to dynamically find out which combination works best for them, instead of designing the “best” interface. The meeting place was chosen to be a paradigm under which the meeting resources are presented to the user—because environment conditions the evolution of small dynamic groups. A workspace is a client visualization of a meeting place and provides an individual view of the place (see Figure 2). It provides a run-time environment for applets and, more generally, for JavaBeans components, providing the basic tools for handling and sharing beans, telepointers, and latecomer support.

A JavaBean is Sun Microsystems’ variation on the idea of a component (Sun Microsystems, 1999a). In object-oriented programming and distributed object tech-

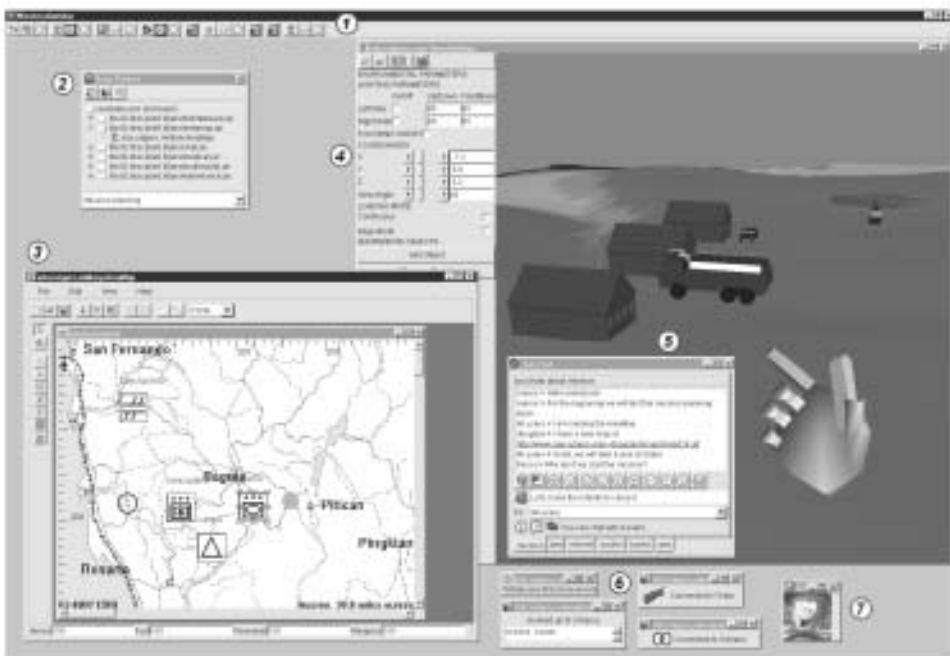


FIGURE 2 Snapshot of a user view during collaboration in a meeting place (1). By using the BeanBrowser tool (2), the user can load beans from the Internet into the BeanBrowser, organize, and import them into the meeting place. The following JavaBeans are present: The AreaMap (3) and ThreeDeeMap (4) are interconnected and display the same geographical area during a military test application. The Chat (5) can show text, pictures, and active URL links and filter messages by chat topic or user name. The multimodal enhancement components (6; multimodal connector, speech, tactile glove, and gaze tracker) and the multimodal manager (7) provide for interface customization.

nology, a component is a reusable program building block that can be combined with other components in the same or other computers in a distributed network to form an application. Examples of a component include a single button in a graphic–user interface, a small interest calculator, or an interface to a database manager. Components can be deployed on different servers in a network and communicate with each other for needed services. Components run within a software environment called a container. Examples of containers include word processors or HTML pages from a Web site displayed in a browser.

Workspaces provide a single seamless UI environment with a high level of consistency. As is the case with the overall desktop view, the workspace also does not require strictly the same view across all clients; each conferee can position the beans at different locations within his or her workspace. However, beans themselves are presently limited to consistent views and actions for all the conferees, because it is not possible to uncover the entire semantics of collaboration-transparent beans (Li, Wang, & Marsic, 1999).

The workspace window is automatically launched as the user enters a place. It shows the beans that are currently in the place (see Figure 2). Sharing a bean for collaboration is as easy as pointing to its URL. A bean can be loaded from a local file system or, given the bean's URL, can be loaded from a Web server. The toolbar on top provides for opening the bean browser (which keeps a list of known beans), changing the telepointer color, and getting help. Workspace awareness involves knowledge of who is present, where they are working, and what they are doing and is used in collaboration to provide appropriate assistance, simplify communication, and manage movement between individual and shared work in a shared workspace. Telepointers, radar views, ticker tapes, messages, and notes (see Figure 3) are provided for this purpose. The toolbar has three buttons for each sharable bean in the workspace: show minimized/hidden, activate/deactivate telepointers, and activate/deactivate radar view.

Telepointers are widgets that allow a given user to track remote users' cursors. Ticker tapes display in the window title bars user messages or membership-related events, for example, when a new user joins the workspace or when somebody leaves. Radar views are widgets that provide global views of the applications along with colored overlays representing each user's local viewing region. They are provided at the bean level because users are free to adjust the size and position of each bean in a workspace and of each workspace on their screens. As shown in Figure 3, the radar view displays the application's dynamic view but also the location of users' viewports in the application frame and the location and motion of users' telepointers. It can be also resized to show a miniature view of the application. A user can filter which telepointers, messages, and notes should be shown in the application frame and radar view.

The users can exchange messages, post small notes, and annotate regions of the bean window to augment their discussion during the meeting. They can also edit their profile (shown in Figure 1) and the properties of the awareness widgets (such as telepointers colors) at any time during the collaboration. The framework also offers several concurrency control algorithms that may be selectively employed (Ionescu, Dorohonceanu, & Marsic, 2000).



FIGURE 3 Collaboration widgets. Three users, *bogdan* (1), *chris* (2), and *boi* (3), collaborate over a chat bean and also post notes. User *chris* can see in his radar view (4) a view of the bean, the notes, telepointers, and viewing areas of all the users.

4. APPLICATION-LEVEL MULTIMODAL INTERFACE

Research has shown that different applications or different application contexts require different human–computer interfaces for best results (Oviatt & Olsen, 1994; Smailagic & Siewiorek, 1996). Some applications do not have deictic data. For example, a keyboard or voice interaction is suitable for a word processing application. On the other hand, pointing devices and gestures are more suitable for an application with spatial content. Multimodality improves naturalness and speed in communication and provides robustness to contexts, users, and tasks. However, there is a tradeoff between having many modalities and the cost and performance of the system. Our goal is again to provide a flexible architecture with a set of modalities for communicating so the user can pick his or her own preferred interaction modes, rather than imposing a single “optimal” multimodal interface for all tasks.

Based on the experience of a previous implementation (Medl, Marsic, Andre, Kulikowski, & Flanagan, 1998), a multimodal fusion and managing agent developed at the Center for Advanced Information Processing was integrated into the DISCIPLE environment (Sletterink, Medl, Marsic, & Flanagan, 1999). The prototype multimodal workstation is illustrated in Figure 4. The implementation focuses on providing modalities for multiple applications in the collaborative environment and providing an easy-to-use library for application developers.



FIGURE 4 The system for multimodal human–computer interaction.

The traditional user interface (keyboard and mouse) was augmented with the following sensory modalities: sound via the Java Speech API (Sun Microsystems, 1999b) and either the Microsoft Whisper speech system (Microsoft Corporation, 1999b) or the IBM ViaVoice (IBM, 1999) speech recognition and synthesis technology, touch via the Rutgers Master II force-feedback tactile glove with gesture recognition software (Burdea, 1996), and sight via the gaze tracker from ISCAN (1998). A multimodal UI management architecture makes these modalities available to the concurrent applications implemented (see Figure 5) for testing. Two main functions of the manager are managing modalities for applications and managing the common fusion patterns. A *fusion pattern* specifies the way modalities are integrated in an application. For application-specific fusion patterns, the manager also may provide common functions to be used by the application developer.

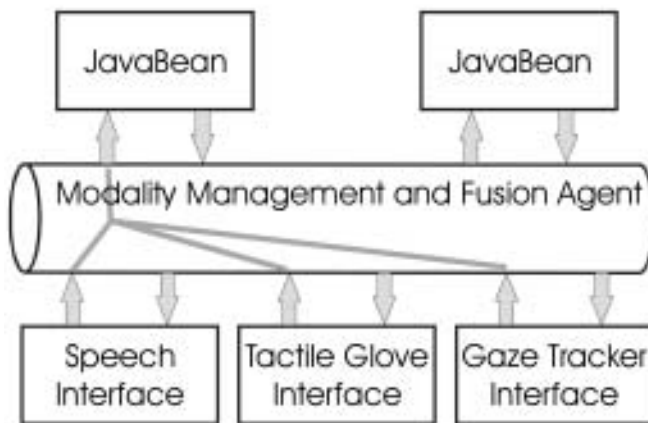


FIGURE 5 Interactions between applications (JavaBeans). The agent and the modalities.

4.1. Modality Manager

The manager interprets multimodal information and handles related problems, such as automatic context- and grammar-switching between applications and hierarchical or selective use of available modalities. The multimodal management system is a virtual modality system, each application not knowing of the existence of other applications that share the modalities. The three main functions are as follows:

- Access and control management for applications to configure modalities
- Route messages between applications and modalities
- Manage modality fusion behavior for different applications

A communication protocol between the multimodal manager and applications helps the applications discover available modalities, receive user commands, and send feedback.

Routing messages from applications to modal outputs is implemented in a straightforward way: They are always sent to all outputs to which they apply. For example, textual feedback will be sent to both the on-screen dialog box and the text-to-speech system, if they are available at the time. For modalities that involve pointing, events are delivered to the window underneath the cursor. The mapping from input to workspace to establish the point of reference within the workspace (the cursor position) may be nontrivial for certain pointing devices. Message routing from nonpointer modalities to applications is done by using the notion of *active application* (selected or focused) in the workspace that currently possesses the user-input focus. Only the active application will get modal input events; for example, recognized phrases from the speech recognizer will be sent to the active application only.

The adaptation of a modality's behavior according to the application that uses it was also addressed. The same notion of active applications also is applied in switching the focus to another application; that is, if another application is selected, the modalities are reconfigured. This means that for the speech recognizer, the recognition grammar and vocabulary are replaced (allowing for more accurate recognition) and for the glove, the look-up table of gestures and the associated actions gets replaced.

4.2. Multimodal Fusion

Fusion of sensory information from multiple modalities can be accomplished at three levels: data, features, or decisions—commands (Sharma, Pavlovic, & Huang, 1998). The fusion agent used is implemented as a variation of the frame-based method, familiar in artificial intelligence practice, and is used in several other multimodal systems (Cohen, Johnston, McGee, Oviatt, & Pittman, 1997; Sharma et al., 1998). It fuses the information at the level of decisions, also called late modality integration. Although some multimodal fusion is application specific, other combinations of actions of specific modalities can be generalized. The manager implements a few of these generic combinations to provide a more

consistent environment for the user and to free the programmer from redundant work. The framework architecture provides for different fusion behaviors. For instance, the glove performs mouse emulation in non-3-D applications but not in 3-D applications, where it represents a virtual hand rather than just a pointer.

The agent fuses low-level input events by keeping a history of events and reusing that data once action-triggering events come in. An action-triggering event can be any event or the last event of a sequence that is recognized to carry semantic information. Examples are sentences coming in from the speech recognizer or the keyboard interface, gestures recognized from the tactile glove, or symbols recognized from the writing tablet.

Actions may not always be properly recognized, or the user may fail to provide all the necessary information. This can lead to insufficient data to perform the recognized command. The system deals with this by responding with an error message.

One of the actions the fusion agent can perform is generating fake mouse events for certain gestures included in the repertoire for the tactile glove. This is called *mouse emulation*. For example, three such gestures were implemented, one for performing mouse clicks, another for performing mouse drags, and yet another for performing normal mouse movement. The corresponding gestures are bending the thumb (as if the user were holding a stick with a button on top), a grasp–hold–release sequence, and a pointing gesture (with the index finger), respectively. These actions work for arbitrary applications. Even if the application is not aware of the multimodal management system, it will still receive mouse events. One might expand the number of gestures with actions like right-clicking or double-clicking, but adding too many gestures would make the recognition less robust and the learning curve steeper for the user.

Controlling the behavior of modalities was implemented as an additional multimodal fusion. An example is control of speech input in three different ways. Background discussions or noise can cause the speech recognizer to accidentally recognize utterances that were never made (an insertion error). Three methods could alleviate this problem: The first method is toggling the activation button on the configuration panel (by mouse). This turns out to be inconvenient. The second method is via spoken system commands, like “disable speech” and “computer listen.” The authors noticed that enabling commands (“computer listen,” “computer enable speech”) are usually not misrecognized from a random conversation. This solution is workable but still imposes a burden on the user. The third method is using *gaze confirmation*, that is, ignoring the speech recognizer when the gaze cursor is off-screen. When the gaze tracker is accurately calibrated and working smoothly, this is a very good solution.

The idea behind gaze confirmation is that people tend to look at the person they are talking to. In this case, the user is looking at the screen. If the user is not looking at the display, the fusion agent assumes that the user is not talking to it, unless the user uses an escaped command (prefixed with “computer”). To avoid frustration with users who have trouble using a gaze tracker (e.g., people wearing hard contact lenses), additional speech commands are implemented to enable or disable gaze confirmation.

Another example of multimodal fusion is switching between applications. Users can employ any combination of speech and gaze. This can be speech only, as in “select whiteboard,” or speech and gaze, saying “select this” and looking at the window, or just staring at the window for a few seconds. The last option is essentially an extension of gaze confirmation; if the user is staring at a certain application, the system assumes intended actions to follow.

4.3. Integration Into the Shared Workspace

The multimodal manager extends the user interface of the workspace itself, the workspace being another application. The manager provides speech commands to create, select, iconify, maximize, or close beans. The pervasive use of simple speech commands and additional modalities helps the user to more quickly become familiar with the multimodal environment. Each participant in a collaborative session may customize his or her interface to interact with the shared application in a way different from the rest of the group. However, these speech commands must always work, regardless of the grammar of the application that currently has the focus. Therefore, the manager dynamically extends the application’s grammar with additional commands. The manager can do so because it is in between the applications and the speech input driver.

Conflicts or ambiguities are possible between the application’s grammar and the manager’s grammar. This, however, is not a problem for the speech recognizer, because it is very well capable of handling redundancies in the grammar. Therefore, no attempt is made to detect this problem while extending and sending the grammar. On the other hand, this is a problem for the parsers. It is handled by requiring the active application to have priority in parsing the sentence. If it fails to recognize the sentence, the manager assumes that it was not meant for the application after all and tries to parse it itself. This might “hide” certain commands for the user. To avoid this, escape commands (commands prefixed with the word *computer*, e.g., “computer, create the Three-Dee-Map bean”) were introduced. Although this makes it impossible for applications to use commands that start with *computer*, the authors believe that this is not a serious limitation. The use of the prefix word is naturally equivalent to the situation where someone wants to attract the attention of another person and calls him or her by name, when it may not be clear who is being addressed.

5. DYNAMIC CUSTOMIZATION OF THE INTERFACE

Componentization of the collaboration framework introduces the possibility for interface customization by the end user. Figure 2 presents an example where multimodal interaction beans are used. They can be loaded as private beans because each user may prefer to augment the workspace with different modalities. By loading and activating different multimodal beans, the user can dynamically select the modalities (e.g., speech, keyboard, eye gaze pointer) for interacting with the workspace. Similarly, by loading different collaboration component beans, the

user can vary the degree of awareness about the other conference participants or select the concurrency control algorithm that applies to a particular bean.

This flexibility allows the DISCIPLE desktop to be used on different types of computers, such as desktop, portable, or wearable computers. This way, people in different environments, or with different computing platforms, can work together. Experiments with the conferees in a heterogeneous collaborative environment with the desktop workstations and wearable computers (see Figure 6), linked with a wireless network, indicate the strong need for dynamically adaptable user interfaces.

The Xybernaut wearable computer is an excellent platform on which to test the claims of portability and interface customization. Although it is basically a shrunken personal computer with a head-mounted display, the user interface as used on a desktop machine is not suitable for this environment. The principle element in collaboration desktop setup is a keyboard (available for desktop use or with a wrist-mounted keyboard). The speech recognizer can make up for a lack of a keyboard; however, dictation software is not always reliable and in fact certain words are never recognized. For example, with a dictation grammar, it is not possible to input the user name "Boi" (mind the *i*) into the workspace login dialog.

Another issue is that the button that replaces the mouse (a round button that can be tilted in any direction to move the cursor, like a joystick) requires user practice. Novice users have a hard time steering the cursor around. They are thus more inclined to use voice commands.



FIGURE 6 The Xybernaut MA IV wearable computer used in collaborative experiments with DISCIPLE.

Finally, the display resolution (640 by 480 pixels) of current head-mounted systems is quite low for modern standards. Most desktop systems nowadays typically have displays 1024 by 768 pixels or higher. The authors observed that, because the collaboration desktop, workspace, and applications were designed on desktop systems, the displayed information is greater than the small head-mounted display can comfortably handle. Heterogeneous computing platforms thus require variations in the presentation layout and application logic, not only in the interaction modalities (Marsic, 2000).

These observations confirm that the system needs to provide flexibility so the user can best adjust the interface to the task needs and the computing environment. The system allows the user at run-time to make interface changes that benefit most applications running within it without requiring modifications of the application code.

6. TEST APPLICATIONS AND EXPERIMENTS

A number of applications have been developed by using DISCIPLÉ. The set of applications includes a whiteboard, a collaborative mapping application, speech signal acquisition and processing, image analysis tools, a military mission planning extension for the whiteboard, and a medical image guided diagnosis system for the diagnosis of leukemia (Comaniciu, Meer, Foran, & Medl, 1998). The military mission planning application allows the user to create and manipulate military units symbolized by icons on a terrain map (Medl et al., 1998). Application development is relatively simple because the developer does not need to worry about the distributed or group issues. The developer simply develops a JavaBean as if a single user will use it, and loading the bean into DISCIPLÉ automatically provides the distribution features. Clearly, the development simplicity cannot be easily measured or entirely attributed to DISCIPLÉ. It is rather a result of choosing JavaBeans and providing a mechanism for automatically making them collaborative.

A chat application and a whiteboard as basic communication tools inside a workspace were implemented for testing. The chat bean (shown in Figure 2) allows for message filtering by chat topic or user name, posting notes and pictures, and active links (in form of URLs) in the text. When users select a URL in a message, the document pointed to by the URL is shown in the default browser of the operating system.

Collaboration work often includes a shared whiteboard, because sketches are a primary means of communicating ideas. The whiteboard allows for simultaneous interaction and image editing and annotation.

Two specialized applications to support two different types of collaborations were used: a mission planning system and an image-guided diagnosis system for blood samples. The mission planning system is designed for military operation planning and disaster relief planning. It involves manipulation of assets on a terrain map, where different icons represent different assets, which can have associated information about resources such as fuel, food, clothing, and other supplies.

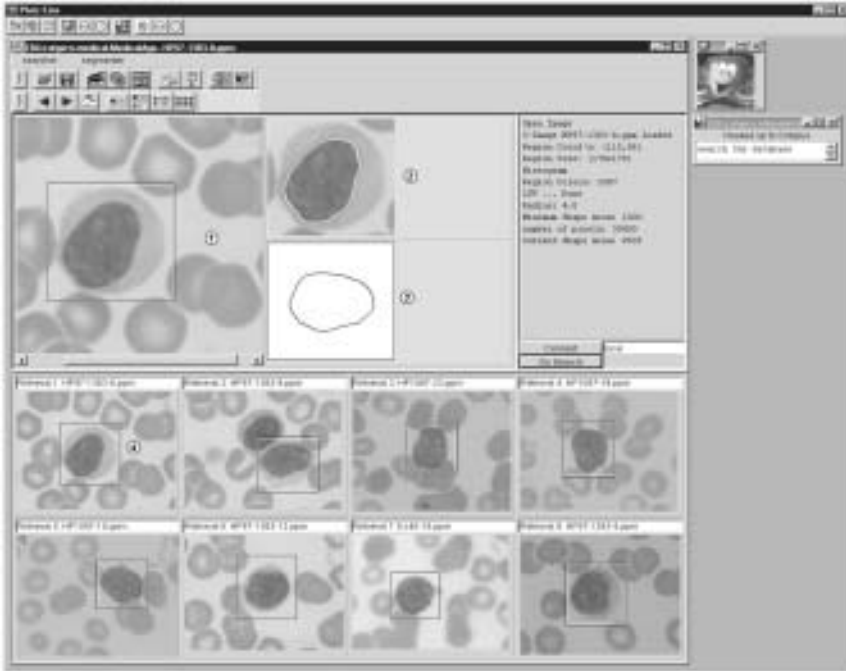


FIGURE 7 The medical diagnosis support application as used in the DISCIPLÉ desktop. The doctor receives an image from a remote microscope and selects a region of interest (1), segments the image (2), selects the cell kernel to extract (3), and searches the database for similar cells (4). Candidates are ranked on similarity.

As shown in Figure 2, there are two different views: a 2-D version, based on the whiteboard framework, and a 3-D version. Both are accessible via the multimodal interface. The 2-D version uses the gaze command more intensively for manipulating icons, whereas the 3-D map uses the tactile glove's 3-D input capability.

The medical application (see Figure 7) uses computer vision techniques to aid doctors in the diagnosis of leukemia (Comaniciu et al., 1998). A blood cell is analyzed based on its size, shape, color, and texture, and a database of different types of leukemia cells and normal cells is searched for closest matches. Because the database contains confirmed diagnoses, the system can provide tentative diagnoses that medical specialists can assess. This application also uses a combination of voice, gaze, and tactile input.

Demonstrations to laboratory visitors were a valuable source of feedback. The reason is that the visitors had no prior knowledge about the system capabilities and thus were more open to explorations. In addition, parallel conversation with the visitors introduced noise and disturbed the flow of interaction with the system. Many times this resulted in random commands being executed, because the background conversation contained phrases that were interpreted by the speech recognition system. Although not strictly supervised and quantified, the demonstrations were much closer to actual everyday use than "clean room" experiments.

6.1. Collaborative Interface Experiments

The authors wanted to investigate how users might use the DISCIPLÉ application framework for customized applications and the degree to which the built-in flexibility helps or hinders collaboration. DISCIPLÉ's user interface provides features previously shown to be useful in collaborative work, such as relaxed WYSIWIS (Stefik et al., 1987) and awareness about group activities (Begole et al., 1999; Gutwin & Greenberg, 1998). Our concern is whether the flexibility to activate or deactivate these features at will positively contributes to collaborative work. In particular, the user has an incentive to use the radar view sparingly because it may affect the application performance on lower end machines. Thus, the participants' actions to activate or deactivate the widgets also were observed in the experiments.

Twelve participants (3 females and 9 males) were recruited from the university student population. All participants indicated they were comfortable using a computer and mouse. Five participants had previous experience with collaborative virtual environments. Potential participants were asked to form their own groups of three before registering to participate. They were not further reassigned to form more or less experienced teams.

Training before the experiment included explaining to the participants the purpose of DISCIPLÉ, how to create a new application, how to load an application into the framework, how to customize the user interface of the framework, and how to use the awareness widgets (Communication Center shown in Figure 1; telepointers, radar views, and notes shown in Figure 3; and messages shown in the ticker tape when posted) provided by DISCIPLÉ during a collaborative session. Next, each participant was asked to load a given simple chat application in DISCIPLÉ (shown in Figure 2), start a 10-min collaboration (chat) session with the other participants, and discover information about them by using the Communication Center and the other awareness widgets.

The study required each team (four teams of 3 participants) to perform a set of three tasks with and without awareness widgets. The decision was made randomly as to whether a team would run the experiment first without the awareness widgets and then by using the widgets or vice versa. When a team used the awareness widgets, they were told that displaying widgets slows down the workstation, so the widgets would better be used only when necessary. Data were collected via three observers who recorded participant behavior in using the awareness widgets and task-related times.

Each team was seated in the same large room. The participants were placed in different cubicles so they could not see each other. They used Windows NT workstations of different computing power connected via an Ethernet LAN. The teams performed the tasks by using the whiteboard similar to the AreaMap shown in Figure 2. The whiteboard was resized to show less than one quarter of the working area. The tasks were modeled after Gutwin and Greenberg (1998) and modified as follows. Each task comprised three subtasks. In each subtask, participants played a different role: User 1, User 2, or User 3. The users were differentiated by the color of their telepointers: blue for User 1, red for User 2, and green for User 3. Each subtask had a maplike background to make the focusing on the foreground

objects more difficult. Verbal communication was not allowed so the participants were forced to use the chat or the awareness widgets (notes or messages).

In Task 1, User 1 randomly specified eight figures of different shape, color, filling, and position as well as which of the other two users (User 2 or User 3) would place each figure. User 2 and User 3 watched for information from User 1 and tried to place figures as soon as fresh information from User 1 became available. In each subtask of this task, the users permuted their roles. The time to accomplish each subtask was measured.

In Task 2, User 1 randomly specified eight rectangles of different color and position. User 2 watched User 1 and placed an ellipse inside each rectangle so that the ellipse followed the rectangular form. User 3 watched and placed a rectangle inside the ellipse placed by User 2. Before placing a new figure, User 1 had to wait for User 3 to finish the job on the previous figure. Again, the users permuted their roles in each subtask. The authors measured the time to construct each triplet object consisting of the three figures.

In Task 3, User 1 randomly specified as fast as possible eight figures of different shape, color, and position. User 2 and User 3 watched for new figures and competed to own them by placing inside them a circle (User 2) or a square (User 3). The user who owned the most figures won. The users permuted their roles in each subtask. The time to accomplish each subtask was measured.

The authors performed one-tailed Student's *t*-tests to see if there were time differences for tasks accomplished with and without awareness widgets. In Task 1 and Task 3, the authors could measure only the time per subtask because User 1 did not have to wait for User 2 and User 3, which may not provide an accurate measure. As can be seen in Table 1, the mean time difference was larger for Task 1, $t(10) = 2.004$, $p < .05$ (one-tailed), $d = 0.81$. The value for sigma in formula for effect size d was computed as suggested in Cohen (1977, p. 44). The authors interpret this as the effect of the radar view (with telepointers) that helped users to faster locate the newly posted requests for figures by following User 1's moves. There was no significant difference for Task 3. The reason may be that the task lasted too short to produce a significant

Table 1: A Comparison of Subtask and Triplet-Object Placement Performance Times (in Seconds) With and Without Awareness Widgets

	Awareness Widgets Allowed	Awareness Widgets Not Allowed	Student's <i>t</i> Test Results	
			<i>t</i>	<i>p</i>
Task 1				
Mean time/Standard Deviation per subtask	402/86	1472/84	2.004	< .05
Number of subtasks	12	12		
Task 2				
Mean time/Standard Deviation per triplet object	42/20	58/21	8.319	< .001
Number of triplet objects	96	96		
Task 3				
Mean time/Standard Deviation per subtask	108/38	123/30	1.109	.140
Number of subtasks	12	12		

time difference. For Task 2, the authors measured the time per triplet object and observed a significant difference, $t(94) = 8.319, p < .001$ (one-tailed), $d = 0.79$. The authors interpret this again as the effect of the radar view as described for Task 1.

On a slow machine, participants preferred to scroll in the whiteboard window and to hide the radar view when they were allowed to do so. On a fast machine, participants preferred to follow the radar view and scroll only when it was necessary. In Task 1, despite the fact that the whiteboard documents provided rulers on the edges for easier positioning, all the participants found posting notes easier than sending messages in chat. In Task 3, all the participants who were allowed or forced to use the radar view, used it to follow the leading user's (User 1) movements.

The authors also examined user satisfaction in a poststudy questionnaire. Participants responded on a scale from 1 (*strongly agree*) to 4 (*strongly disagree*) to the questions shown in Table 2.

On average, DISCIPLINE framework was easy to understand and use. The communication center was not very helpful, but this is normal, because its role is only to provide static information about the users. Participants did not use the telepointers very much. This could be due to the fact that users preferred to see the telepointers in the radar view, instead of seeing them in the application frame, considering that showing the telepointers in both the radar view and application would be redundant. When participants had to perform the tasks without awareness widgets, some of them complained that they had to scroll too much in the whiteboard frame.

Instead of typing long messages in chat, participants preferred short messages (in chat or ticker tape) combined with telepointers and notes. This combination is natural and more expressive, because in face-to-face communications we use gestures along with speech to include spatial information. The communication via messages was moderate in Task 1, high in Task 2, and none in Task 3. Messages were mostly used to communicate subtask completions. Communication via notes was high in Task 1 (User 1 used them to specify the exact positions where objects should be placed), sporadic in Task 2 (sometimes User 1 and User 3 tried to help User 2), and none in Task 3. In summary, awareness tools were more likely to be used in the map-based, gamelike (construction, competition), and dynamic tasks.

Table 2: User Perceptions of Collaboration Using DISCIPLINE Framework and Its Awareness Widgets

<i>Question</i>	<i>Mean score</i>
I found DISCIPLINE framework easy to understand.	1.75
I found DISCIPLINE framework easy to use.	1.92
I could perform the tasks better using some of the awareness widgets.	1.58
I found the communication center helpful in accomplishing the tasks.	3.00
I found the chat application helpful in accomplishing the tasks.	2.00
I found the telepointers helpful in accomplishing the tasks.	2.50
I found (DISCIPLINE's) radar view helpful in accomplishing the tasks.	1.50
I found the notes helpful in accomplishing the tasks.	1.42
I found the messages helpful in accomplishing the tasks.	2.00

Note: The lower the mean score, the more satisfactory the user's experience.

Face-to-face interaction normally includes many feedback functions and behaviors, such as conversation initiation and giving and taking turns (Cassell, 2000), which are difficult to convey in a distributed collaborative environment. Group awareness components aim to remedy this problem. However, experiments of the authors and other researchers demonstrate that usual group awareness widgets are insufficient. The users lack many of the face-to-face cues to verify that their actions are understood. Many times users required additional confirmation (“Can you see what I did?”) or offered unsolicited notification (“I just did this.”). Conveying user behaviors across the network is a major open problem in groupware systems. Issues related to user embodiment, such as nonverbal communications, facial expression, and involuntary movement, are being investigated through the use of sophisticated avatars that can provide such user embodiment (Capin, Pandzic, Thalmann, & Thalmann, 1998; Snowdon & Tromp, 1997). Although this work is groundbreaking with respect to achieving the goal of virtual worlds completely mimicking the real world, it requires sophisticated virtual reality software and hardware. One of the authors’ current efforts is in using a gaze tracker to communicate the user’s focus of attention to the collaborators.

6.2. Multimodal Interface Experiments

Initial performance experiments for the multimodal interface compared the combinations of two modalities at a time for the same task and captured the times it took for users to perform the tasks (Marsic et al., 2000). The command used in the evaluation was “Create <object> <at location>” on a situation map, such as in Figure 2. The location reflects the variation in the combination of modalities and can be given by speech, for example, “create camp at Baguio,” or a location pointed to by the glove or gaze. For example, “speech and speech” means that both the object and the location are specified by speech. Preliminary performance data for the total command execution time over 10 trials for 2 participants (one male, one female) are shown in Figure 8. The combination of speech and gaze has a definite speed advantage over the other modality combinations. The location precision for icon creation appears to be greatest for speech-and-speech because the coordinates are exactly specified. Among pointer-based modalities, precision appears to be better for mouse than for gaze or glove. The speech-and-gaze combination is the most lacking in precision.

Experiments with the multimodal interface show that different combinations of modalities afford speed versus accuracy in interacting with the applications. The modalities presented here provide more flexibility and functionality than keyboard and mouse, but they are primitive at this stage of development and need extensive refinement. These experiments also show that there is a learning curve for using these modalities even though they are assumed to be natural for human communication. Probably the greatest limitation is the simplistic language understanding component and the knowledge-based intelligent fusion capabilities. The current set of voice commands is simply not powerful enough to cover a broad range of interactions. Although the individual modalities are still immature, significant gains will be possible

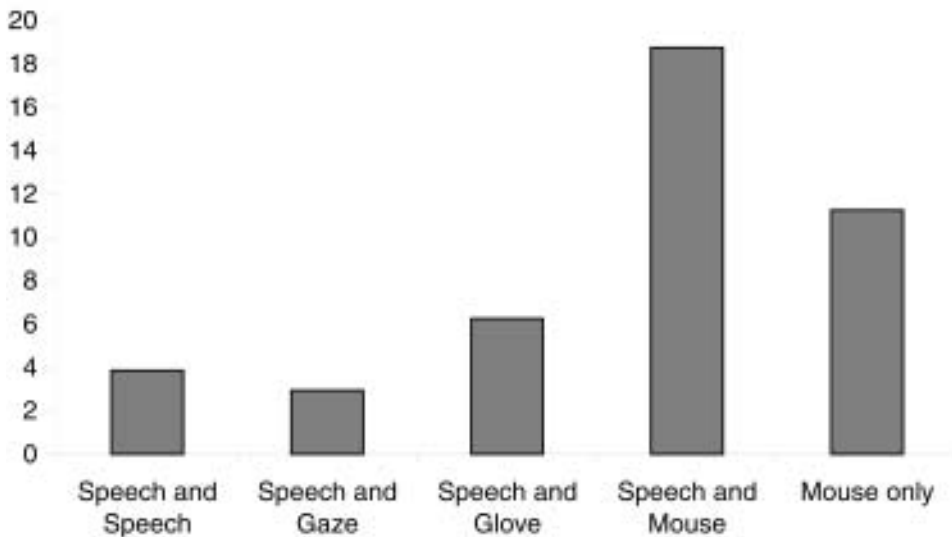


FIGURE 8 A comparison of total execution times for interface modality combinations for a command: "Create <object> <at location>." Vertical axis shows time values in seconds, given to the closest millisecond (Marsic et al., 2000).

by increasing the intelligence of the system, in particular by improving the command interpretation module and knowledge representation of the application.

7. CONCLUSIONS

This article presents the design and implementation of a framework for dynamically adaptable user interfaces for telecollaboration. The framework provides a run-time environment for JavaBeans components and allows interaction with the components, distributes their events remotely, and provides mechanisms to control cooperative features in an application-independent manner. Other unique features of the DISCIPLE framework include simultaneous support for collaboration-aware and collaboration-transparent applications and the ability to support ad hoc collaboration groups.

The framework itself is developed by using JavaBeans. Componentization of the collaboration framework introduces the possibility for interface customization by the end user. The interface is open and customizable with respect to the context in which it is placed and particular user needs. By loading and activating beans with different functionalities, the user can dynamically customize the collaborative application. This flexibility allows the DISCIPLE desktop to be used on different types of computers, such as desktop, portable, or wearable computers.

The application framework approach presented here has advantages over the commonly used toolkit approach. With toolkit approaches, the application designer makes decisions about the application functionality, whereas in this ap-

proach the end user makes the decisions to meet the actual needs of the task at hand.

As developed to date, DISCIPLER has been implemented and tested on both third party collaboration-unaware beans and the authors' collaboration-aware beans. The authors' experience with these applications reinforces the need for flexible user interfaces. There is a need on one hand to increase the user's awareness about the group activities and to widen the communication channel between the human and the computer as well as between the collaboration participants. On the other hand, overly complex interfaces become unmanageable and cumbersome to use, while at the same time requiring extra resources. Software components recently have gained great interest as a means for making customizable applications. Our approach exploits this flexibility and extends it to customization of user interfaces. Rather than imposing an "optimal" user interface, UI componentization allows the users to tailor the UI to their particular needs and to the task at hand.

Updated information and a free source of the DISCIPLER software can be downloaded from the Web site at <http://www.caip.rutgers.edu/disciple>

REFERENCES

- Beca, L., Cheng, G., Fox, G. C., Jurga, T., Olszewski T., Podgorny, M., et al. (1997). Web technologies for collaborative visualization and simulation. In M. T. Heath (Eds.), *Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing* (CD-ROM). Philadelphia: Siam.
- Begole, J. B., Rosson, M. B., & Shaffer, C. A. (1999). Flexible collaboration transparency: Supporting worker independence in replicated application-sharing systems. *ACM Transactions on Computer-Human Interaction*, 6(2), 95-132.
- Bentley, R., Rodden, T., Sawyer, P., & Sommerville, I. (1992). An architecture for tailoring cooperative multi-user displays. In J. Turner & R. Kraut (Eds.), *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW '92)* (pp. 187-194). New York: ACM.
- Burdea, G. (1996). *Force and touch feedback for virtual reality*. New York: Wiley.
- Capin, T. K., Pandzic, I. S., Thalmann, D., & Thalmann, N. M. (1998). Realistic avatars and autonomous virtual humans in VLNET networked virtual environments. In J. Vince & R. Earnshaw (Eds.), *Virtual worlds on the Internet* (pp. 157-173). Los Alamitos, CA: IEEE Computer Society.
- Cassell, J. (2000). More than just another pretty face: Embodied conversational interface agents. *Communications of the ACM*, 43(4), 70-78.
- Cohen, J. (1997). *Statistical power analysis for the behavioral sciences* (Rev. ed.). New York: Academic Press.
- Cohen, P. R., Chen, L., Clow, J., Johnston, M., McGee, D., Pittman, J., et al. (1996). Quickset: A multimodal interface for the distributed interactive simulation. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '96)—Demonstration Session*. New York: ACM Press.
- Cohen, P. R., Johnston, M., McGee, D., Oviatt S., & Pittman, J. (1997). QuickSet: Multimodal interaction for simulation set-up and control. In *Proceedings of the 5th Applied Natural Language Processing Meeting*. Washington, DC.
- Comaniciu, D., Meer, P., Foran, D., & Medl, A. (1998). Bimodal system for interactive indexing and retrieval of pathology images. In K. M. Hanson (Ed.), *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision (WACV '98)* (pp. 76-81). Princeton, NJ: IEEE Press.

- Dorohonceanu, B., & Marsic, I. (1999). A desktop design for synchronous collaboration. In A. K. Peters (Ed.), *Proceedings of Graphics Interface '99 (GI '99)* (pp. 27–35). Kingston, ON.
- Flanagan, J. L., & Marsic, I. (1997). Issues in measuring the benefits of multimodal interfaces. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)* (pp. 163–166). Piscataway, NJ: IEEE Press.
- Gellersen, H. W. (1995). Modality abstraction: Capturing logical interaction design as abstraction from “user interfaces for all.” In C. Stephanidis (Ed.), *Proceedings of the 1st ERCIM Workshop “User Interfaces for All”*. Retrieved April 17, 2003 from <http://ui4all.ics.forth.gr/UI4ALL-95/proceedings.html>
- Gutwin, C., & Greenberg, S. (1998). Effects of awareness support on groupware usability. In J. Coutaz & J. Karat (Eds.), *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '98)* (pp. 511–518). New York: ACM.
- IBM. (1999). ViaVoice technology for speech recognition and synthesis. Retrieved April 17, 2003, from <http://www-3.ibm.com/software/speech>
- Ionescu, M., Dorohonceanu, B., & Marsic, I. (2000). A novel concurrency control algorithm in distributed groupware. In H. R. Arabnia (Ed.), *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '2000)* (Vol. 3, pp. 1551–1557). Las Vegas, NV: CSREA Press.
- ISCAN. (1998). RK-446 single target video tracking system. Retrieved April 17, 2003, from <http://www.iscaninc.com>
- Krasner, G., & Pope, S. (1988). A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. *Journal of Object-Oriented Programming*, 1, 26–49.
- Li, W., Wang, W., & Marsic, I. (1999). Collaboration transparency in the DISCIPLE framework. In S. C. Hayne (Ed.), *Proceedings of the ACM International Conference on Supporting Group Work (GROUP '99)* (pp. 326–335). New York: ACM.
- Malone, T. W., Lai, K.-Y., & Fry, C. (1992). Experiments with Oval: A radically tailorable tool for cooperative work. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW '92)* (pp. 289–297). Toronto, ON: PUBLISHER.
- Mansfield, T., Kaplan, S., Fitzpatrick, G., Phelps, T., Fitzpatrick, M., & Taylor, R. (1999). Toward locales: Supporting collaboration with Orbit. *Journal of Information and Software Technology*, 41, 367–382.
- Marsic, I. (1999). DISCIPLE: A framework for multimodal collaboration in heterogeneous environments. *ACM Computing Surveys*, 31(Elec. Suppl. 2).
- Marsic, I. (2000). Real-time collaboration in heterogeneous computing environments. In B. Werner (Ed.), *Proceedings of the IEEE International Conference on Information Technology: Coding and Computing (ITCC 2000)* (pp. 222–227). Piscataway, NJ: IEEE Press.
- Marsic, I., & Dorohonceanu, B. (1999). An application framework for synchronous collaboration using Java beans. In R. H. Sprague (Ed.), *Proceedings of the Hawaii International Conference on System Sciences (HICSS 32)* (pp. XX–XX). Piscataway, NJ: IEEE Press.
- Marsic, I., Medl, A., & Flanagan, J. L. (2000). Natural communication with information systems. *Proceedings of the IEEE*, 88, 1354–1366.
- Medl, A., Marsic, I., Andre, M., Kulikowski, C. A., & Flanagan, J. L. (1998). Multimodal man-machine interface for mission planning. In *Proceedings of the AAAI Spring Symposium on Intelligent Environments* (pp. 41–47). Stanford, CA: AAAI Press.
- Microsoft Corporation. (1999a). NetMeeting 3.0. Retrieved April 17, 2003, from <http://www.microsoft.com/netmeeting>
- Microsoft Corporation. (1999b). Windows Highly Intelligent Speech Recognizer: WHISPER. Retrieved April 17, 2003, from <http://www.research.microsoft.com/research/srg/whisper.htm>
- MITRE Corporation. (1999). Collaborative virtual workspace (CVW). Retrieved April 17, 2003, from <http://www.mitre.org/pubs/showcase/cvw.html>

- Oviatt, S. L., & Olsen, E. (1994). Integration themes in multimodal human-computer interaction. In *Proceedings of the International Conference on Spoken Language Processing* (pp. 551-554). Tokyo, Japan: Acoustical Society of Japan.
- Roseman, M., & Greenberg, S. (1996). Building real-time groupware with GroupKit, a groupware toolkit. *ACM Transactions on Computer-Human Interaction*, 3, 66-106.
- Sharma, R., Pavlovic, V. I., & Huang, T. S. (1998). Towards multimodal human-computer interface. *Proceedings of the IEEE*, 86, 853-869.
- Sletterink, B., Medl, A., Marsic, I., & Flanagan, J. L. (1999). *Multimodal user interface management*. Student poster presented at the 8th HCI International Conference, Munich: Germany.
- Smailagic, A., & Siewiorek, D. (1996). Matching interface design with user tasks: Modalities of interaction with CMU wearable computers. *IEEE Personal Communications*, 3, 14-25.
- Snowdon, D., & Tromp, J. (1997). Virtual body language: Providing appropriate user interfaces in collaborative virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST '97)* (pp. 37-44). New York: ACM.
- Sun Microsystems. (1999a). JavaBeans specification. Retrieved April 17, 2003, from <http://www.javasoft.com/beans/docs/spec.html>
- Sun Microsystems. (1999b). Java Speech API for incorporating speech technology into user interfaces, applets, and applications. Retrieved April 17, 2003, from <http://www.javasoft.com/products/java-media/speech>
- Stefik, M., Bobrow, D. G., Lanning, S., & Tatar, D. G. (1987). WYSIWIS revised: Early experiences with multiuser interfaces. *ACM Transactions on Information Systems*, 5, 147-167.
- Syri, A. (1997). Tailoring cooperation support through mediators. In W. Prinz, T. Rodden, H. Hughes, & K. Schmidt (Eds.), *Proceedings of the 5th European Conference on Computer-Supported Cooperative Work (ECSCW '97)* (pp. 157-172). Lancaster, England: Kluwer Academic.
- W3C Architecture Domain. (1999). Extensible Markup Language. Retrieved April 17, 2003, from <http://www.w3.org/XML>
- Usability 1st. (1999). Retrieved April 17, 2003, from <http://www.usabilityfirst.com>
- Winiwarter, W. (1996). VIENA classroom—The efficient use of natural language within CSCW hypermedia environments. In *Proceedings of the Australasian Natural Language Processing Summer Workshop*. Retrieved December 1, 1999 from <http://www.ifs.univie.ac.at/~ww/anlp96.html>