Identity-Differentiating Widgets for Multiuser Interactive Surfaces

Kathy Ryall, Alan Esenther, Clifton Forlines, Chia Shen, and Sam Shipman
Mitsubishi Electric Research Laboratories

Meredith Ringel Morris
Stanford University

Katherine Everitt
University of Washington

Frédéric D. Vernier
University of Paris

By applying the iDwidgets concept, the authors supplement traditional widgets with identity differentiation that supports widget reuse, dynamic widget customization, clutter reduction, and novel multiuser widget type creation. This article introduces a conceptual framework for iDwidgets, describing four axes that application can customize by exploiting identity differentiation: function, content, appearance, and group input.

Widgets—standard reusable GUI elements—are a staple of user-interface development. The use of widget toolkits, such as Java's Swing, X Window System's Motif, or Microsoft's MFC, allows programmers to quickly incorporate a number of standard interactions (such as clicking buttons, selecting check boxes, or scrolling through lists) into their software. To date, most widgets have been designed for use by one person at a time. Within a single session, a widget will behave the same regardless of who uses it. However, traditional single-user widgets don't support cooperative work systems with multiple users.

Pebbles[1] and The MultiDevice Multi-User Multi-Editor (MMM)[2] were two of the first systems to extend widgets for computer-supported cooperative work. They extended the visual representations of widgets to distinguish among users in shared-display settings. In a short paper published at Interact-2005, we introduced the notion of identity-differentiating widgets (iDwidgets) for collaborative settings.[3] iDwidgets extend the widget concept by including identity as an input parameter, which lets us customize interactions in a variety of ways. In that paper, we presented a few hypothetical examples to illustrate the concept.

Here we expand the iDwidget presentation, providing example implementations of iDwidgets from each of four categories—function, content, appearance, and group input—that we can customize by identity differentiation. We have incorporated these sample widgets into several tabletop groupware systems across a number of domains and usage scenarios. By providing multiple, concrete examples of iDwidgets in action, we hope to show the utility of the conceptual framework, and provide other application developers with ideas for exploiting identity in tabletop and other group settings.

iDwidgets

iDwidgets are basic GUI building blocks for user-aware environments, such as multiuser interactive tables. The iDwidget's novelty is that its function, contents, and/or appearance can be parameterized by the identity of its current user among a group of copresent (local or remote) users. Thus, an iDwidget might look or behave differently for different user identities. By identity, we mean a person with particular preferences and privileges, or a tool associated with such a person (for example, the stylus the person is using). A single person might have multiple identities (for example, dad and senior engineer). Multiuser tabletops and other forms of shared-display groupware are the target use setting for iDwidgets. These spaces differ from the traditional desktop in that they must support multiple users who might share an interaction surface. This paradigm introduces user interface challenges such as access control, preference optimization, reach, and surface clutter. Interactive tables have been a motivating scenario for our development of the iDwidgets concept and prototypes.

Our goal for iDwidgets is to increase a widget's utility and support widget reuse. A single widget instance serves multiple people, helping reduce clutter in shared-display groupware applications. In cases where widget replication might have advantages, iDwidgets don't preclude replication. In essence, iDwidgets are a mechanism to provide support for, and extend the notion of traditional widgets to multiuser settings. They also enable new widget creation. The questions of how best to use these widgets or how to choose among alternative widgets are still open research questions.

We have developed a number of sample applications using iDwidget prototypes. Figure 1 shows a few examples, illustrating the contexts in which the applications are used. Subsequent figures with screen shots reveal

 Published by the IEEE Computer Society

1 **Tabletop iDwidgets in action: (a) DTMap, (b) TeamTag, (c) UbiTable, and (d)Table-for-*N*. Each application uses a MERL DiamondTouch table, which serves as an interactive tabletop.**

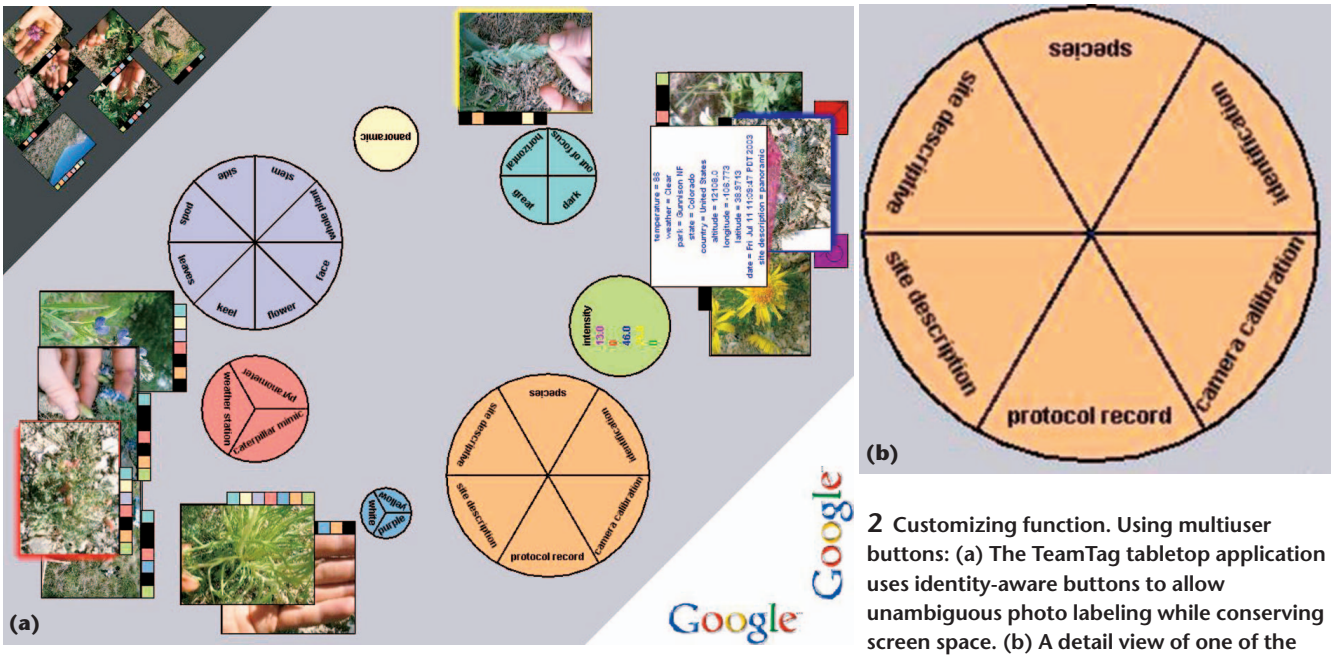
more application details later in this article. We created these example applications for a Mitsubishi Electric Research Laboratories (MERL) DiamondTouch table,[4] a multiuser tabletop surface that accepts simultaneous touch input from up to four people. The table determines identity by determining which conductive seat pad a user is sitting on. Capacitive coupling from the table to the seat pad via the user's body enables the table to determine which user is touching at which location. However, iDwidgets are not limited to the DiamondTouch or other capacitive identity-sensing technologies. Any system that can provide identity information can take advantage of the iDwidgets concept. Many ubiquitous computing environments (for example, face recognition, biometrics, RFID) exploit identity-differentiating technology to build applications for multiuser environments. We propose embedding the identity information at a lower level than the application or system level by encapsulating it into reusable iDwidgets.

Conceptual framework and example implementations

The ability to identify a widget user is the key feature that enables a system to identify iDwidgets. User identity becomes a parameter to the widget. In general, iDwidgets can exploit any attributes or behaviors that are customizable in traditional (single-user) computing environments. Rather than creating multiple tool bars or other controls (one per person), a system can use a single instance of an iDwidget, providing flexible and customizable interaction for different people, and potentially providing space savings. We can make iDwidget versions of many traditional widgets by extending and customizing them along any of four dimensions: function, content, appearance, and group input. New widgets are possible as well. Table 1 summarizes the four dimensions for customizing widgets based on identity information.

Table 1. Summary of the four dimensions of widget customization.

Customization Dimension	Conceptual Examples	Sample Implementations
Function: behavior varies	Multiuser buttons, semantic interpretation, differentiated behavior, privileged access	Team Tag, PebblesDraw, DTMap, AudioNotes, UbiTable
Content: content varies	Menus, lists	Table-for-*N*, PebblesDraw
Appearance: looks vary	Properties, aesthetics, orientation	E-Poetry, DTMap, SoundTracker
Group input: supports and/or requires interaction from multiple people	Cumulative effect, simultaneous input, modal input sequences, logging and audit trails	Sides, DTMap, SDG ColorMixer Control



2 Customizing function. Using multiuser buttons: (a) The TeamTag tabletop application uses identity-aware buttons to allow unambiguous photo labeling while conserving screen space. (b) A detail view of one of the identity-aware labeling buttons.

Our motivation and rationale for defining the iDwidgets frameworks is rooted in both single-user WIMP environments and our own more recent work in shared tabletop applications. In its simplest form, a user profile on a traditional single-user system implements our notion of customizable iDwidgets. When John logs on to a computer, his personal profile populates the desktop and dictates the appearance of GUI elements; similarly, when Sara logs on to the same machine, her user profile helps customize the session for her. If John uses the computer while Sara is logged in, however, he is presented with her bookmarks (when using a Web browser), her address book (when using an email application), and her file access permissions (when trying to access documents). In a multiuser system, iDwidgets would give both John and Sara their own, customized environment using a single set of widgets. Thus, we derived the first three dimensions of our framework (function, content, and appearance) from traditional user-profile or resource files. As WIMP interfaces are used in group settings, however, a richer set of interaction techniques becomes possible. Our fourth dimension, group input, captures this new widget category.

We believe the four dimensions of customization that our framework defines are complete; any customization based on identity information will fall within one of these categories. A widget is defined by its look and feel. Our four categories capture both of these dimensions. We have divided the look of a widget into appearance and content, while dividing its feel into function and group input. The conceptual and implemented examples we describe in this article are intended to illustrate each of the dimensions—they are not a comprehensive list of all possible instantiations of the iDwidgets concept. By identifying the framework and providing examples of each type of customization, we hope to inspire others to explore the possibilities. In the following sections, we explain how each of these four dimensions can exploit identity knowledge and provide example iDwidgets that we have developed in the course of our research on tabletop groupware.

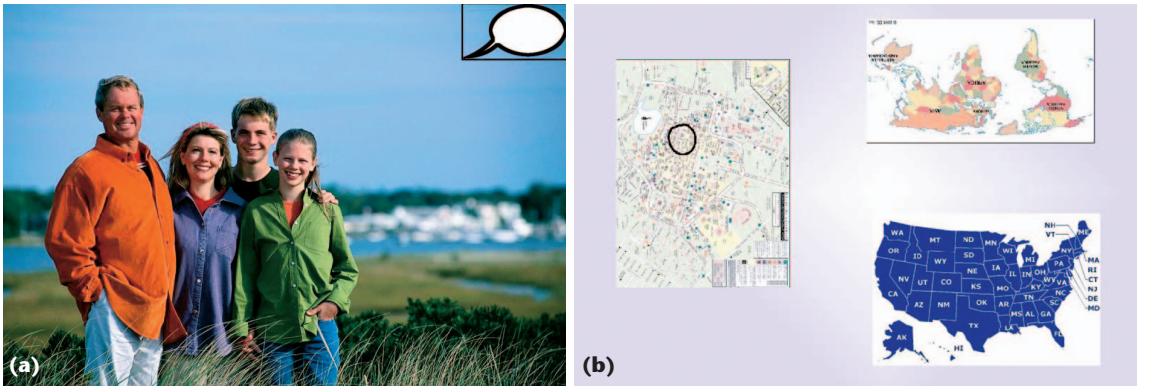
Customizing function

We can customize an iDwidget's function by passing along user-identity information to the widget's event handler. The widget looks the same to all users, but behaves differently depending on who interacts with it. We have explored several techniques for customizing widget functionality based on identity, including multiuser buttons, semantic interpretation, differentiated behavior, and privileged access.

Multiuser buttons

Multiuser buttons are an extension of traditional, single-user buttons that perform a different function based on who has clicked them. DTMap (see Figure 1a) includes a multiuser undo button. When pushed by a particular person, it causes the last action of that person to be undone, rather than the last global action to be undone. The application keeps track of the multiple undo stacks, and the user-identity parameter indicates which stack to access.

TeamTag (see Figure 1b and Figure 2) provides another example of multiuser buttons. TeamTag is a tabletop application that lets user groups associate metadata with their digital photos; each user has an active photo associated with her (that is, the photo that she is currently labeling).[5] The (multiuser) labeling buttons each represent a particular category of labels (for example, people or locations), and subdivisions within the widget represent specific values for that category (for example, Bob or Sue). To minimize clutter onscreen, only one copy of each button appears in the application. However, because the buttons are identity aware, the correct photo label pairings occur, even when multiple users simultaneously interact with the same buttons.



3 Customizing function. (a) Using semantic interpretation. The speech bubble icon on this photo from the AudioNotes system is an identity-aware caption widget that allows personalized variants of photo captions to be available to different users. (b) Using differentiated behavior. The flashcards in this educational tabletop application deliver level-appropriate hints to different students.

Semantic interpretation

Semantic interpretation is another manner in which we can customize widget function with identity information. A single string (or graphic) might denote different objects for different users. For example, "dad" is a different alias for most people, only identifying the same person for siblings. In a photo-browsing application with a search feature, if John entered "dad" in the search box, the system would return pictures of his father; if Mary entered "dad," it would return pictures of her father.

Another example of customizing semantic interpretation based on user identity is the facility for customizing photo captions in our AudioNotes application. AudioNotes is a tabletop application that allows groups of users to share digital photographs. These photos can have captions associated with them, as indicated by the speech bubble iDwidget (see Figure 3a). When a user touches the bubble, a personalized caption is displayed to him privately, either through individually targeted audio[6] or through a message displayed on a personal auxiliary device, such as a PDA or laptop. For instance, when Alma touches the caption widget on a photo of her family, she hears "dad, mom, and me at the Mission Beach," while her friend Fred hears "Mr. and Mrs. Reyes and Alma at Mission Beach." Currently, system users create personalized captions manually; better face recognition from photographs combined with information from personal address books could help automate the creation of such captions in the future.

Differentiated behavior

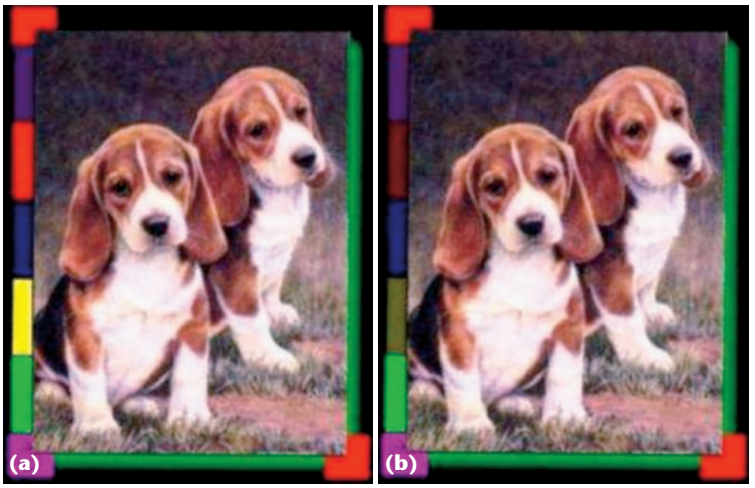
Differentiated behavior is yet another approach to customizing widget function on a per-user basis. A single instance of a widget performs the same action, but behaves differently based on the user's identity. For example, a scroll bar might provide continuous scrolling for one user and discrete scrolling for another, based on their prespecified preferences, or a paintbrush might vary in its numerical values of brush thickness range (for example, thin, medium, or broad) on a per-user basis. We have instantiated differentiated behavior to provide the appropriate level of help in an educational tabletop activity, where groups of students interact with digital flashcards (see Figure 3b). The flashcards themselves embody a hint-giving widget, and, when touched, a hint is delivered to the touching user via individually targeted audio. The system customizes the level of hint delivered (easy, medium, or hard) on a per-user basis based on a configuration file (intended to be edited by the teacher) that reflects each user's current level of material mastery. It would also be possible to customize the hints as specified by a student's earlier answers and interactions, creating a widget that provides dynamic differentiated behavior not only based on which user was interacting with it, but also on that user's history.

Privileged access

Privileged access is another possible functional customization of widgets. A particular widget might only work for some special subset of users, and might not respond when users outside of this privileged group interact with it. A security widget, for example, might respond only to a senior member of the group touching it. Privileged access also supports document ownership in shared spaces. In our work on multiuser coordination policies for shared-display groupware, we developed a widget for granting and revoking access permissions to documents on a tabletop on the fly (see Figure 4 on the next page).[7] This widget demonstrates the concept of privileged access: only the user who owns the document in question can grant and deny access to others—by touching the colored tabs on the document's side that correspond to the colors of the other users' chairs. When other group members who do not own the document touch the colored tabs, their input is ignored. In UbiTable (see Figure 1c) we also exploit user identity to support privileged access.[8] While anyone can manipulate any document on the table, only the owner of a document may copy or move it back to a personal device. Document ownership is assigned based on the device where the document originates or when the owner passes the document to a colleague's space.

Customizing content

User-differentiated widget content is also a rich area for customization. In this case, when a specific user acti-



4 Customizing function. With privileged access, the colored tabs along the edge of this photo are widgets for on-the-fly permission granting and revocation. Only the user who owns the image (in this case, the green user) can successfully interact with these widgets. Touching the colored tabs toggles permissions for other group members. (a) In this photo, green, yellow, and red users have permission. (b) If the green user touches the red and yellow tabs, then a new state results, revoking the permissions for those two users.

vates the widget, the widget's contents will be different for different people, potentially providing different options via custom lists or menus.

Lists

The iDwidget version of a traditional list widget varies the list contents depending on who is accessing the list. For example, a history list in a Web browser on a shared display might vary depending on the currently interacting user. The displayed list's contents might be generated on the fly depending on who is interacting with the widget. Lists of bookmarks could also be populated in real time depending on user identity. We have explored customized bookmarks lists in the Table-for-*N* application[9] that incorporates the personalized views multi-user coordination policy.[7] Table-for-*N* is a tabletop application that allows a user group to share and annotate various types of media, including text, HTML, photos, and video. Users can switch between several virtual tabletops (analogous to the concept of virtual desktops for PCs) and can bookmark items for easy access between virtual tabletops. The bookmarks list will dynamically adapt its content depending on which user is currently touching the widget. Thus, the items listed in the widget vary.

Menus

Menu iDwidgets can also have their contents customized based on user ID. Customized menus are in many ways similar to traditional pull-down menus whose contents are the same across users, but access to certain menu items (or submenus) can be determined based on who is interacting with the menu. Menu items may also be reordered (for example, most recently used items by a particular user appear closer to the top), or inactive (for example, grayed out) depending on who is using the menu. Certain menu items could even be removed on a per-user basis (in which case the custom menu takes on similar functionality to the custom list). The coordination policy augmented Table-for-*N* application (see Figure 1d) customizes the contents of shared menu bars based on user identity by showing the user's current menu settings highlighted in gray. In this case, the listing of options and actions is the same for each user, but the toggle indicating his or her current selection varies.

Customizing appearance

The application can customize a widget's appearance (for example, fonts, colors, and so on) using identity information without necessarily changing its function or content. For instance, a widget's properties and aesthetics, or its orientation, are potentially useful surface features to customize on a per-user basis.

Properties and aesthetics

Properties and aesthetics such as colors, fonts, languages, and other traditional graphical features of a widget can be customized based on the user, while the widget's functionality remains unaffected. While some might argue that label language on a widget might impact its functionality, we prefer to think of it as a user preference or usability issue. Adapting some of the widget properties (for example, large type) is especially important for elderly or disabled individuals. In Figure 5a, we show an electronic magnetic poetry application in which identity information determines which language to display on the label of a word tile, in this case either English or Japanese.[9] In Figure 5b, we show the annotation feature from DTMap in which a user's pen color and thickness is customized using identity information.

Orientation

A widget's orientation might also be customized based on identifying information, which is especially relevant for horizontal displays. Combining user identity with location information would allow widgets to dynamically orient themselves to a particular user, which can increase legibility as well as signal ownership and usage. Automatic handle positions might also be determined by user identity and location. Our SoundTracker application[6] customizes orientation in this manner—all objects in the system (such as photos and song clips) automatically turn to face the touching user (see Figure 5c). Because the DiamondTouch associates identity with a particular conductive chair pad, it's feasible to assume a correspondence of user identity to a specific seating configuration.

Customizing group input

iDwidgets can support or require group input, enabling interaction from multiple people. We distinguish these iDwidgets from previous multi-input widgets (for example, two-handed input), which are typically used by one person. These single-user multi-input widgets, could, of course, also be extended to exploit the identity information provided to an iDwidget. Possibilities for customizing group input via identi-

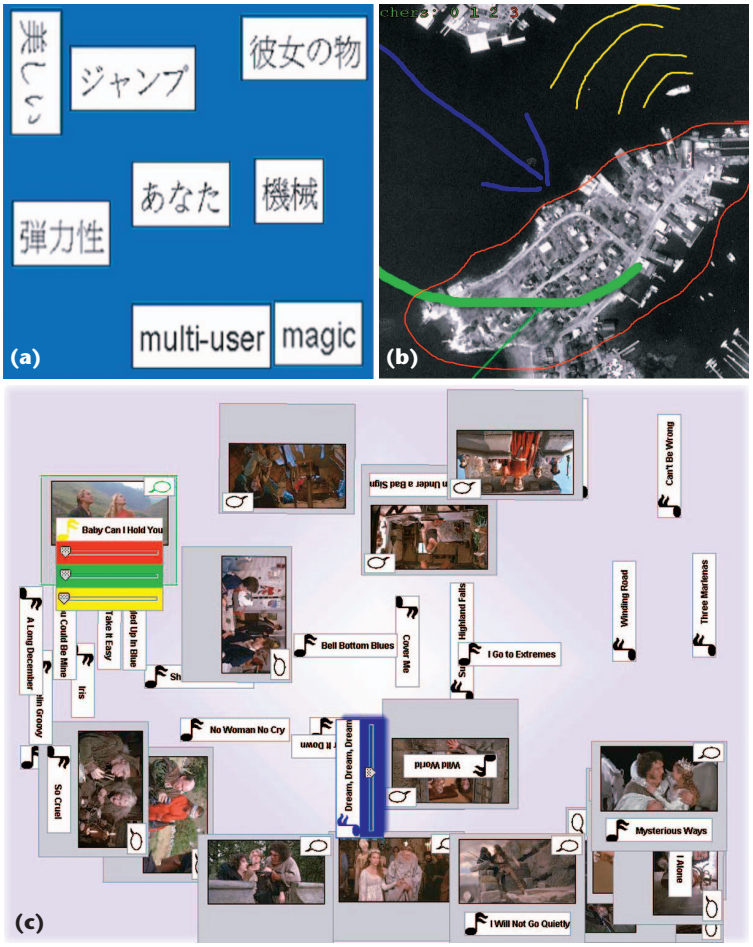
ty information include widgets having a cumulative effect, requiring simultaneous input, supporting modal input sequences, and having built-in audit and logging.

Cumulative effect

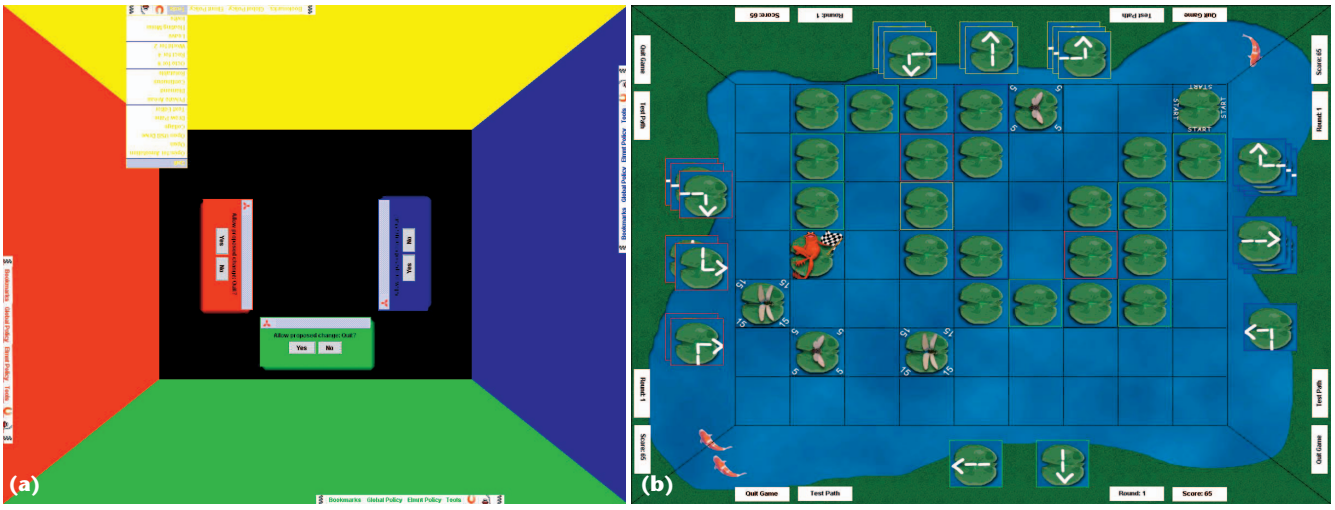
Cumulative-effect iDwidgets require interaction from a number of people before taking some action. A voting widget, for example, might require all users to respond before tallying the result. As an alternative, it might only require a quorum to agree, without requiring everyone to respond. The number of distinct users interacting with the widget (as indicated by the identity information) would be the distinguishing factor. In our work on coordination policies, we implemented a user-aware voting widget (see Figure 6a). When an application-wide change (such as clearing the screen) is selected from a menu by one user, the other users in the group are each presented with a voting widget allowing them to vote for or against the proposed action. Identity information prevents users from voting with others' widgets and ensures that all users' votes are received and factored in to the decision.

Simultaneous input

Simultaneous-input widgets add a strict time constraint on top of a cumulative-effect widget, by requiring synchronized action by multiple users to activate a widget. Two people, for example, might be needed to turn virtual silo keys to launch a missile. Other examples include large surface interactions, where one person cannot directly reach all the needed objects or regions. In a variant of the Shared Interfaces for Developing Effective Social Skills (Sides) system (developed at Stanford University), special buttons are located on the four edges of the table, and the four system users must simultaneously touch them to make key changes in game state, such as testing a solution to this puzzle game (see Figure 6b). In addition to creating heightened group awareness of key system events (one of several goals of cooperative gesturing interactions[10]), simultaneous input in Sides aims to serve a therapeutic



5 Customizing appearance. (a) For language choices in an e-poetry application, English is displayed when one user touches a tile, Japanese when a different user touches. (b) For graphics customization, the graphical properties of each user's pen, in this case color and line thickness, are customized based on identity. (c) Orientation is customized in SoundTracker as items on the table turn to face the user who touched them.



6 Group input. (a) Using cumulative-effect widgets, the yellow user selects the option to quit the application from a menu in Table-for-*N*. The other three users are each presented with identity-aware voting widgets. A unanimous vote is required for the quit action to proceed. (b) For simultaneous input, in a variant of the Sides game, the buttons replicated on each edge of the table must be simultaneously activated by the four users as a means of stimulating increased conversation and coordination among the users.

Related Work

Although many toolkits[1-3] have been developed for multiuser spaces, most only provide simultaneous access to traditional, single-user widgets, and don't provide any new widgets. A notable exception is DiamondSpin,[2] which provides multiple toolbars and support for per-widget reorientation; it has recently incorporated many iDwidgets concepts. The SDG Toolkit[3] provided the first group input widget we have seen referenced in the literature. Its SDG ColorMixer Control behaves differently depending on which and how many people are interacting with it. This toolkit raises and solves many general multiuser system issues, and would make an excellent platform to implement a larger set of iDwidgets. Finally, while tool-based systems might provide support for multiple people to work in parallel, each tool is not inherently user aware; a particular tool behaves the same regardless of who is using it. While some provide support for group input, they don't necessarily exploit user differentiation information. Benford et al.[4] created two systems for collaborative storytelling. KidPad and Klump are a nice illustration of SDG widgets in action. Of particular note, Klump[4] provides a group input mechanism that allows multiple people to interact collaboratively and synchronously control a single object.

Some aspects of iDwidgets are present in a number of other systems. MMM[5] was perhaps the earliest system containing pieces of the iDwidget concept. Its multiuser environment supported multiple people working together on a single, shared display. It supported a shared mouse (used serially by different users) that worked with a particular user's defaults and preferences until the device was registered to another user. While its goal to support per-user commands, modes, and preferences is in the same spirit as iDwidgets, its use of identity information seems to be more at the device and application level; it's not clear that they incorporated it at the widget level. We propose embedding the identity information into the building blocks for an application, encapsulating it into reusable iDwidgets, rather than at the application or system level.

Pebbles[6] was the first system to add user ID information to widgets. In particular, PebblesDraw included a multiuser undo button (separate undo stacks for each user), custom menus (graying out inappropriate selections on a per-user basis), and privileged access (controlling and limiting access to a widget based on identity and/or number of people interacting with the widget). Much of this system's focus, however, is on the visual representation needed to distinguish people (and their actions and interactions) in shared-display settings. The iDwidgets concept further exploits user identity information, extending and generalizing the concept to encompass a larger set of functionality; it provides a conceptual framework in which to think about the use of identity information to customize interaction.

References

1. H.P. Hourcade and B. Bederson, *Architecture and Implementation of a Java Package for Multiple Input Devices (MID)*, Univ. of Maryland Human–Computer Interaction Lab. (HCIL) tech. report no. 99-08, 1999.
2. C. Shen et al., "DiamondSpin: An Extensible Toolkit for Around-the-Table Interaction," *ACM Conf. Human Factors in Computing Systems* (CHI), ACM Press, 2004, pp. 167-174.
3. E. Tse et al., "SDG Toolkit: Rapidly Prototyping Single Display Groupware through the SDG Toolkit," *Proc. 5th Australasian User Interface Conf.*, Australian Computer Soc., 2004, pp. 101-110.
4. S. Benford et al., "Designing Storytelling Technologies to Encourage Collaboration between Young Children," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems* (CHI), ACM Press, 2000, pp. 556-563.
5. E. Bier and S. Freeman, "MMM: A User Interface Architecture for Shared Editors on a Single Screen," *Proc. ACM Symp. User Interface Software and Technology* (UIST), ACM Press, 1991, pp. 79-86.
6. B. Myers, H. Stiel, and R. Gargiulo, "Collaboration Using Multiple PDAs Connected to a PC," *Proc. Conf. Computer Supported Cooperative Work* (CSCW), ACM Press, 1998, pp. 285-294.

role in stimulating increased conversation and coordination from the system's target user group—for example, adolescents with Asperger's syndrome.

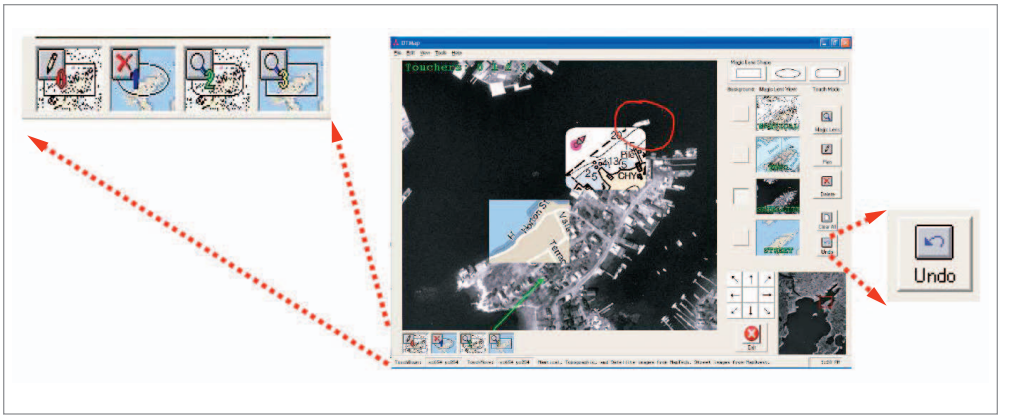
Modal input sequences

Modal input sequences provide another opportunity for iDwidget customization: enabling iDwidgets to support parallel moded interactions. By tracking identity information, widgets can support the interleaving of different people's actions. One person can be in delete mode while another person is simultaneously in annotate mode. With traditional widgets, once the delete mode is activated, the next object touched would be deleted. With iDwidgets, only the next touch of the person in delete mode would delete an object. In the example screen shot shown in Figure 7, four users are interacting simultaneously: user 0 is annotating, user 1 is deleting, and users 2 and 3 are manipulating Magic Lenses. User 2 has selected a nautical view revealing depth information, while user 3 holds a street map view. This form of customization can increase an application's efficiency and utility; we have used this capability extensively in most of our tabletop applications. The parallel nature of tabletop and other collaborative settings make parallel-interleaved modal input sequences one of the most critical iDwidget capabilities.

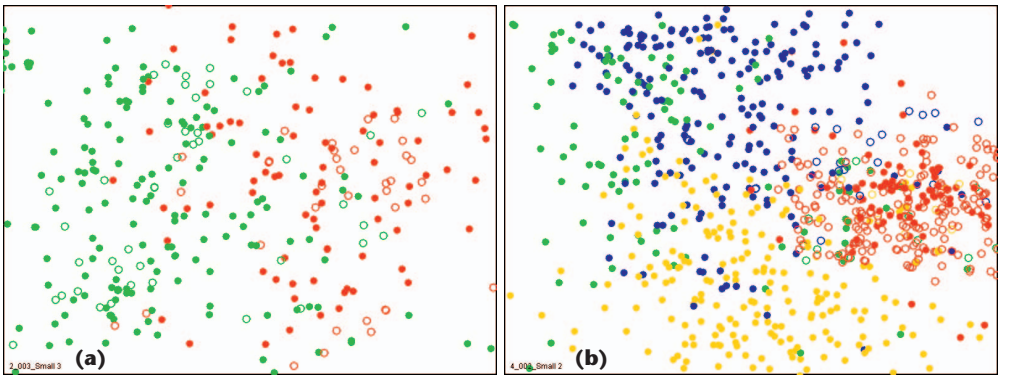
Logging and audit trails

iDwidgets also support easy logging and audit trail creation. Because an iDwidget knows who is touching, the identification information can be added to the log file. Many other systems provide such multiuser audit support (such as Microsoft Word's tracked changes feature); however, iDwidgets incorporates it at the widget level. In Figure 8 we show a visualization of two user-study sessions developed using iDwidgets. Because the iDwidgets exploit user identity, they can easily generate a log file. This example shows the user ID and action type for each touch to the table. The log file might also include other information such as touch duration, event

type (for example, touch, release, and drag), or change in touch (for example, gesture transition). Visualization itself is not part of the iDwidgets framework. However, exposing and providing appropriate logging data suited for visualization, audits, and playback is one of the important capabilities of iDwidgets.



7 Group input. DTMap supports modal input sequences. The legend (located along the bottom left edge of the application and shown in larger detail on the left) indicates each user's mode. DTMap also illustrates multiuser buttons; the undo button (lower right) is parameterized by who activates it, undoing that person's last action, rather than globally undoing the action performed by the last user to touch the table.



8 Group input. Example visualization of a sample tabletop user-study session showing logging and audit trails: (a) a two-person session and (b) a four-person session. Color indicates which user touched the table (red for user 1, green for user 2, blue for user 3, and yellow for user 4); fill indicates action type (solid for touching one type of widget and outline for touching a second type of widget).

Discussion and future work

The power of a widget lies in encapsulating a set of behaviors and packaging them along with graphical attributes so that it can easily be used and reused. The iDwidget concept—adding user identity as a parameter to customize a widget in a multiuser setting—enables interactions with widgets to be dynamically adapted on a per-user basis for group usage. In addition to the benefits of personalized interactions, iDwidgets support widget reuse and sharing.

As current hardware matures and new technology becomes available, the power of iDwidgets will increase as well. Moreover, many of the problems addressed by iDwidgets become more significant on bigger tables with more simultaneous users. This holds true for tabletop settings, and more generally for any multiuser interactive surface or collaborative group setting in which identity differentiation is available as a feature. For example, DTControls[11] showcases a novel technology for physical widgets that support user-identity differentiation. The iDwidgets interaction paradigm could easily be used in that and other similar settings.

As noted earlier, iDwidgets provide a mechanism for widget reuse, and for multiuser customization. When and how it is appropriate to allow shared widgets, and which customization dimension(s) should be used are two application-dependent questions, and should be left as a policy question for developers. While some fragments of the iDwidget concept are present in a number of other systems, iDwidgets promotes identity to a first-class widget parameter, providing a unifying framework for specifying the customization of shared widgets.

iDwidgets also raise new feasibility and usability issues. Which widgets lend themselves to identity differentiation, and which do not? What happens when two people simultaneously access a menu—do one user's preferences take priority, or do we temporarily replicate the menu? What effects will dynamically adapting widgets have on interactions with an application? When will customizing a widget enhance performance, and when will it hinder learnability?

iDwidgets represent a generalization of ideas already in practice today. The enabling technology is already available and in use, most notably in tabletop and ubiquitous computing systems; many applications already incorporate fragments of the iDwidgets idea. By extending and generalizing the use of identity in widgets introduced in earlier work and moving the identity information out of the application or system level and into the widgets, iDwidgets provides a conceptual framework in which to think about the use of identity information to customize interaction, and opens the door for new application development and new lines of research. ■

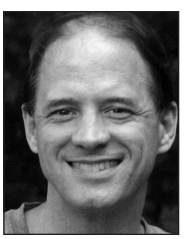
Acknowledgments

We acknowledge our collaborators whose research and development efforts contributed to some of the example systems discussed in this article, especially Anne Marie Piper and Eileen O'Brien (Sides), Chalmers Wang (AudioNotes), and Dan Morris (SoundTracker).

References

1. B. Myers, H. Stiel, and R. Gargiulo, "Collaboration Using Multiple PDAs Connected to a PC," *Proc. Conf. Computer Supported Cooperative Work* (CSCW), ACM Press, 1998, pp. 285-294.
2. E. Bier and S. Freeman, "MMM: A User Interface Architecture for Shared Editors on a Single Screen," *Proc. ACM Symp. User Interface Software and Technology* (UIST), ACM Press, 1991, pp. 79-86.
3. K. Ryall et al., "iDwidgets: Parameterizing Widgets by User Identity," *Proc. Interact*, Springer Verlag, 2005, pp. 1124-1128.
4. P. Dietz and D. Leigh, "DiamondTouch: A Multi-User Touch Technology," *Proc. ACM Symp. User Interface Software and Technology* (UIST), ACM Press, 2001, pp. 219-226.
5. M.R. Morris et al., "TeamTag: Exploring Centralized versus Replicated Controls for Co-Located Tabletop Groupware," *ACM Conf. Human Factors in Computing Systems* (CHI), ACM Press, 2006, pp. 1273-1282.
6. M.R. Morris, D. Morris, and T. Winograd, "Individual Audio Channels with Single Display Groupware: Effects on Communication and Task Strategy," *Proc. Conf. Computer Supported Cooperative Work* (CSCW), ACM Press, 2004, pp. 242-251.
7. M.R. Morris et al., "Beyond "Social Protocols": Multi-User Coordination Policies for Co-Located Groupware," *Proc. Conf. Computer Supported Cooperative Work* (CSCW), ACM Press, 2004, pp. 262-265.
8. C. Shen, K. Everitt, and K. Ryall, "UbiTable: Impromptu Face-to-Face Collaboration on Horizontal Interactive Surfaces," *Proc. UbiComp*, Springer Verlag, 2003, pp. 281-288.
9. C. Shen et al., "DiamondSpin: An Extensible Toolkit for Around-the-Table Interaction," *ACM Conf. Human Factors in Computing Systems* (CHI), ACM Press, 2004, pp. 167-174.
10. M.R. Morris et al., "Cooperative Gestures: Multi-User Gestural Interactions for Co-Located Groupware," *ACM Conf. Human Factors in Computing Systems* (CHI), ACM Press, 2006, pp. 1201-1210.
11. P.H. Dietz et al., "DT Controls: Adding Identity to Physical Interfaces," *Proc. ACM Symp. User Interface Software and Technology* (UIST), ACM Press, 2005, pp. 245-252.

Kathy Ryall *is on the principal technical staff at MERL. Her research interests include human–computer interaction, computer-supported cooperative work, and information visualization. Ryall has PhD in computer science from Harvard University and is a senior member of the IEEE. Contact her at ryall@ merl.com.*



Alan Esenther *is a user interface software developer in the Technology Lab at MERL. His research interests include human–computer interaction; finding the right digital tool for the job; and making multiuser, multitouch, and large touch surface interactions and development easier. Esenther has an MS in computer engineering from Boston University. Contact him at esenther@merl.com.*

Clifton Forlines *is a researcher at MERL. His research interests include the design and evaluation of novel user interfaces, digital video presentation, collaborative tabletops, multiuser and multidisplay workspaces, and using handheld projectors for augmented reality. Forlines has an MS in human–computer interaction from Carnegie Mellon University. Contact him at forlines@merl.com.*

Chia Shen *is a senior research scientist at MERL. Her research interests include human–computer interfaces and interactions with nonconventional visual surfaces. Shen has a PhD in computer science from the University of Massachusetts at Amherst. Contact her at shen@merl.com.*

Sam Shipman *is a principal member of the technical staff at MERL. His research interests include real-time analysis of video and audio content, and real-time and distributed operating systems. Shipman has a BS from the University of North Carolina-Wilmington and an MS in computer science from Carnegie Mellon University. Contact him at shipman@merl.com.*

Meredith Ringel Morris *is a researcher in the Adaptive Systems and Interaction group at Microsoft Research. Her research interests include human–computer interaction, computer-supported cooperative work, and educational technologies. Morris has a PhD in computer science from Stanford University. Contact her at merrie@ stanfordalumni.org.*

Katherine Everitt *is a PhD student in computer science at the University of Washington. Her research interests include tabletop interfaces, computer-supported cooperative work, and multimodal interfaces. Everitt has an MS in computer science from the University of California, Berkeley. Contact her at everitt@cs.washington.edu.*

Frédéric D. Vernier *is an assistant professor in the Department of Computer Science, University of Paris Sud in France, where he works in the LIMSI-CNRS laboratory. His research interests include human–computer interaction and interactive visualization. Vernier has a PhD in computer science from the University of Grenoble. Contact him at frederic.vernier@limsi.fr.*