Heterogeneous Collaboration Using XML

Ivan Marsic[†], Allan Krebs[†], Helmuth Trefftz[†], Bogdan Dorohonceanu[†], and Marilyn Tremaine[‡]

[†]CAIP Center
Rutgers University
Piscataway, NJ 08854-8088 USA
+1 732 445 6399
{marsic, krebs, trefftz, dbogdan}@caip.rutgers.edu

[‡]College of Information Science and Technology
Drexel University
Philadelphia, PA 19104-2875 USA
+1 215 895 2490
tremaine@acm.org

ABSTRACT

The recent proliferation of computing devices and use contexts demand equivalent diversity in collaborative applications. Our work on the DISCIPLE and Manifold frameworks supports the development of collaborative applications for these heterogeneous environments. Using eXtensible Markup Language (XML) for the communication medium provides for the heterogeneity. Collaborators share the same data or a subset of the data, represented in XML, but view very different displays depending on their computing capabilities. Mobile users, for example, may view a more limited budget presentation than their collaborating colleague in the office. The Manifold framework transforms information to match the client's local capabilities and resources, yet maintains important content information. We used DISCIPLE and Manifold to implement a 3D to 2D collaborating environment and measured team performance characteristics in a simple object placement task.

Keywords

Heterogeneous computing, collaboration frameworks, mobile computing, groupware, CSCW.

INTRODUCTION

A key component for synchronous collaboration is real-time sharing and manipulation of information. Most collaborative applications provide synchronization support, the so-called WYSIWIS (What You See Is What I See) technique [21]. WYSIWIS allows one user to point or discuss an object on the screen that they are confident is visible to their collaborator. Providing support for WYSIWIS with past systems was a difficult but not impossible task because the groupware used common platforms. Developing groupware applications that are interoperable across diverse environments is significantly more difficult and costly. However, with recent anytime-anywhere proliferation of computing technology, support for heterogeneity is inevitable. Mobile applications involving synchronous collaboration are emerging in many fields, e.g., business, healthcare, military services and transportation. The classic example is the provision of just-in-time assistance between a deskbound expert and a mobile fieldworker using a portable device. The mobile worker may work with blueprints while the expert is using a 3D CAD model to repair a vehicle or the plumbing in a building. Kraut *et al.* [13], for example, show that fieldworkers make quicker and more accurate repairs when a remote expert is providing assistance.

The heterogeneity of computing platforms manifests itself in CPU speed, memory, display capabilities, and network bandwidth, with the last two accounting for the most prominent differences. This paper focuses on display differences since current developments indicate that they are likely to be the most variable. For example, we are seeing the development of displays mounted in eyeglasses[1], OptiScape technology[2], windshield-projected displays in cars, or foldable displays[3]. These displays are invariably more limited in size and quality than those on the desktop. Enlarging the screen real estate for mobile computers remains impractical because of the inconvenience of weight and size. Weight may also limit the compute power and bandwidth needed to display the graphics and animation that occurs in a desktop environment. Moreover, there is also a human information processing limitation that constrains the mobile user. A person in a private office can allocate vastly different cognitive resources to a display than a person in the field who needs to pay attention to the external environment, e.g., traffic [11].

All the above factors result in information displays that are both smaller and more limited in display capabilities for the mobile user than the stationary user. The problem then becomes one of enabling collaboration between users who have different views of the same information within an application and even run different applications.

---

[1] http://www.microopticalcorp.com/

[2] http://www.inviso.com/

[3] http://www.ices.cmu.edu/design/

We take a *data-centric approach*, where conferees share the same data or a subset of that data. Our DISCIPLE system [23] provides a container for importing Java applications and making them collaborative. Our Manifold framework defines a set of applications that use XML[4] for data exchange on heterogeneous devices.

We demonstrate the Manifold framework via a collaborative session involving office furniture arrangement between users with different display capabilities. Some of the users are viewing a 3D representation of the office, while other users are seeing the same office in two dimensions. In addition to providing an instantiation of our heterogeneity support, we also have evaluated the effect of the heterogeneous environment on the quality of the collaboration.

The paper is organized as follows. We first review related work in this area. Then we overview the architecture of the DISCIPLE system and describe the XML-based Manifold framework for heterogeneous collaboration. Then we present a short study that compares heterogeneous environment collaborations to homogenous environment collaborations. Finally, we discuss our results from this comparison and conclude the paper.

BACKGROUND AND RELATED WORK

The need to allow conferees to collaborate on dissimilar terminals was recognized early on by D. Engelbart, the pioneer of computer-supported collaborative work [5]. Many collaborative systems and toolkits address these same issues. For example, Garfinkel *et al*. [7] discusses the adaptation of the shared display (in particular, color maps) to various display devices. If necessary, their system, SharedX, degrades the quality of the displayed images to match the capabilities of the display hardware. However, the authors assume that users have the same replicated application and are just applying device-specific rendering.

An early design for heterogeneous groupware is presented in [12], but it does not employ a model-view separation. Rendezvous [17], GroupKit [19] and several groupware toolkits thereafter use model-view separation so that developers can create models and drive different views. However, no implementation is attempted, and in some cases (e.g., [17]) the situation is greatly simplified by using centralized groupware architecture.

A recent approach to collaboration in heterogeneous computing environments is the CMU Pebbles project [15]. Pebbles is focused on single-display groupware, with the team being in a single meeting room. Multiple handheld personal digital assistants (PDAs) provide simultaneous input (mouse, keyboard) to a single workstation. PDAs are not treated as equal partners in the collaboration, which belies the need for the heterogeneous data representation.

The Visage system from Maya Design [8] is a powerful (single-user) visualization system for creating custom visualizations and direct manipulation of large and diverse datasets. While Visage addresses diverse data visualization with polymorphic views, it is not explicitly intended for collaboration in heterogeneous computing environments. Recent work on Visage Link [14], an extension of Visage, adds multiuser collaboration capabilities, but Visage Link does not consider heterogeneous models.

An important aspect of heterogeneity is interoperability between the groupware systems. Dewan and Sharma [4] address this issue. Another approach [16] is grounded on the premise that almost any information can be encoded as 2D pictures, and that humans readily understand such pictures. The focus is on a common pictorial surface that contains a rendering of graphical objects. Sharing between collaborators is implemented at the level of the pictorial surface (i.e., view), with PostScript as the communication standard. As the authors point out, editing the surface is cumbersome since application logic is lost in the rendering process. Our work is complementary to these efforts insofar as we focus on an architecture for developing new applications for heterogeneous environments, rather than interoperating the existing ones.

Using XML for data exchange is key to our architecture for heterogeneous groupware. XML from its inception was intended to separate data representation from visualization. To accomplish this, data are described in the XML document whereas the rendering is determined by an XSL (eXtensible Style-sheet Language) document. Several efforts are underway to realize XML's potential for information exchange on heterogeneous devices. The Wireless Application Protocol (WAP) Forum has standardized Wireless Markup Language (WML), an XML language optimized for specifying presentation and user interaction on limited capability terminals such as mobile phones, pagers, two-way radios, and smartphones [24]. Several companies, e.g., Nokia, already offer WAP-compliant devices, but this architecture currently does not utilize XML/XSL separation. WearLogic[5] developed an XML data model and browser for a credit-card-size personal information organizer. It implements simple data rendering, but not arbitrary applications. To our best knowledge, there is currently no other research effort that uses XML/XSL separation for dynamic run-time adaptation of application models and visualizations to diverse devices.

In addition to developing an architecture for heterogeneous collaborative applications, we evaluate team performance characteristics for such applications. Although the WYSIWIS idealization recognizes that efficient reference to common objects depends on a common view of the work at hand, strict WYSIWIS was found to be too limiting and relaxed versions were proposed to accommodate personalized screen layouts [21]. In subsequent work [22], problems were reported with non-WYSIWIS systems because manipulation and editing processes were private

[4] http://www.w3.org/XML/. See also http://www.xml.com/

[5] http://www.wearlogic.com/

and only results were shared. This discontinuity of the interaction created ephemeral environment differences that affected collaboration.

Our essential principle for heterogeneous collaboration is that every user's action is interactively and continuously reflected in other users' workspaces, with a varying degree of accuracy or realism or through a qualitatively different visualization. Thus, the shared reference is maintained. We should also point out that non- WYSIWIS is quite common in collaborative virtual environments (CVEs) [9,20] where collaborators navigate independently to accomplish their own goals. A recent study [9] suggests that users have difficulties in establishing mutual orientations in CVEs, but that having some common frame of reference, for example a 2D map added to the CVE, might alleviate this. Another study [1] found that asymmetries in collaborative interfaces impair collaboration, but the impact is decreased if the asymmetries matched the roles.

SYSTEM ARCHITECTURE

DISCIPLE forms the communication infrastructure for collaboration. It is a mixture of client/server and peer-to-peer architecture and is based on *replicated architecture* for groupware [3]. Each user runs a copy of the collaboration client and each client contains a local copy of the applications that are the foci of the collaboration.

Figure 1 shows the architecture of the DISCIPLE system. The set of conferees is represented hierarchically as an Organization, and they meet in Places. The central part of DISCIPLE is conceptualized as the collaboration bus [23]. The bus achieves synchronous collaboration through real-time event delivery, event ordering and concurrency control. The collaboration bus comprises a set of named communication channels, where the peers can subscribe to and publish information. The channels may aggregate the events to reduce the network traffic and processing load.

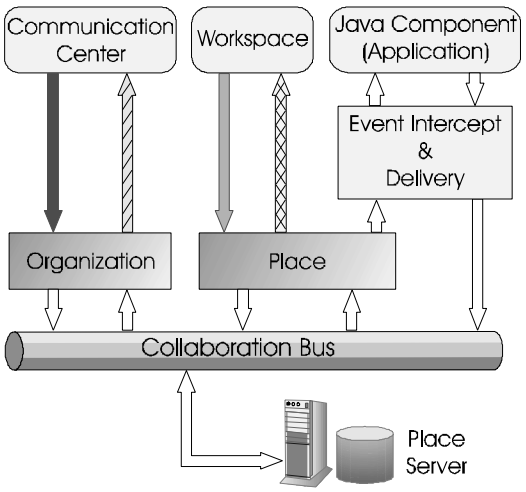

Figure 1: DISCIPLE architecture. Organizations and Places are abstractions implemented as multicast groups. They are represented in the user interface as Communication Center and Workspaces, respectively.

Sharing Java Beans

DISCIPLE is an *application framework*, i.e., a semi-complete application that can be customized to produce custom applications. End users perform customization at runtime select and import task-specific Java Beans[6] for their collaboration support. The DISCIPLE workspace is a *shared container* where Beans can be loaded very much like Java Applets downloaded to a Web browser. Collaborators import Beans by drag-and-drop manipulation into the workspace. The imported Bean becomes a part of a multi-user application and is available to all conferees. Objects in the Bean are not aware of distributed services or sharing. They interact with DISCIPLE through the Beans event model as with any other event source/listener. DISCIPLE handles the distribution. The application framework approach has advantages over the commonly used toolkit approaches. With toolkits the application designer decides about the application functionality. With our approach, the end user makes the decisions. We consider the latter to be a better solution because it allows each user to adapt the collaboration to the specific task.

MANIFOLD BEANS

DISCIPLE provides several features that handle heterogeneity primarily at the communication/networking level, but not at the application logic and interaction levels. For this, the application Bean, itself, must be properly "architectured." We present a class of Beans called ***Manifold Beans***, which are designed to deal with heterogeneity at these levels.

The main characteristic of the Manifold beans is the multi-tier architecture. The common three-tier architecture comprises the vertical tiers of presentation, application or domain logic, and storage. The Manifold's *presentation* tier is virtually free of the application logic and deals with visualizing the domain data and accepting the user inputs. The *domain* tier deals with the semantics of tasks and rules as well as abstract data representation. The third tier in our case comprises the collaboration functionality, which is mainly provided by the DISCIPLE framework, but a part of it resides in the bean as discussed below.

The key concept is a Glyph, which represents all objects that have a geometry specification and may be drawn [2]. Figure 2 shows a simplified class diagram. All Glyphs in a document form the scene graph, itself a Glyph, which has a tree data structure. The scene graph is populated with different vertices (Glyphs) in different applications that extend the Manifold framework. Glyphs are divided into two groups. Leaf Glyphs represent individual graphic elements, such as images, geometric figures, text or numbers in spreadsheet cells. PolyGlyphs are containers for collections of Glyphs. They correspond to branch nodes and can have children. Example PolyGlyphs are group figures, paragraphs, maps or calendars. PolyGlyphs have

[6] http://www.javasoft.com/beans/

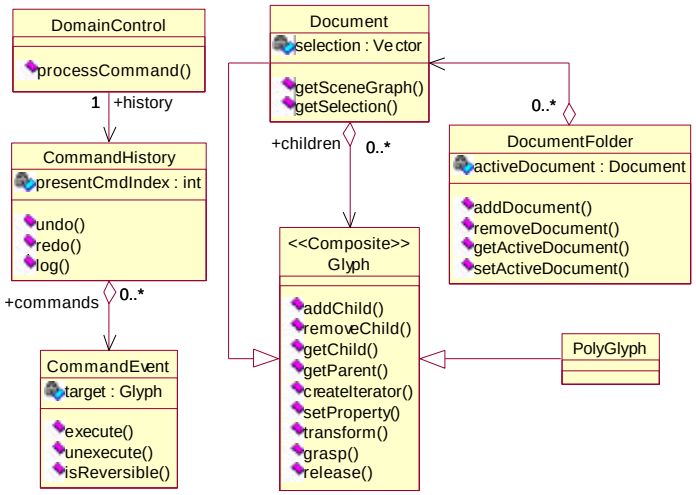

Figure 2: Domain class diagram of a generalized editor.

all the functionality of the Glyphs and the additional property that they can contain Glyphs or other PolyGlyphs.

The operations on the Glyph tree are: create Glyph (Op_1), delete Glyph (Op_2), and set Glyph properties (Op_3). The corresponding Glyph methods are `addChild()`, `removeChild()`, and `setProperty()`. The method `transform()` is essentially a `setProperty()` method, but we introduce it to speed-up execution by avoiding checking if the property name is "transformation" every time we alter the property. The method is likely to be called very frequently during direct manipulation of a glyph. Properties include color, dimensions, constraints on glyph manipulation, etc.

Glyphs are sources of the following types of events which are fired in response to the operations: AppearanceEvent for add/remove operations, PropertyChangeEvent, and TransformEvent. The interested parties register as event listeners for some or all types of the events via the Java Beans delegation event model.

DomainControl is the system controller for the domain bean that invokes the system operations. This is the only portal into the domain bean. The only way to cause a state change in the bean is to invoke the `processCommand()` method. Even the local presentation (view) objects interact with the domain objects through this portal only.

The CommandEvent class implements the Command pattern [6] and has the responsibility of keeping track of the argument values to invoke operations on Glyph and Document objects so the operations can be undone/redone. We name this class CommandEvent instead of Command to emphasize the Java event distribution mechanism. Commands create/delete Glyphs or correspond to Glyph methods. In addition, we have commands to open or save a document and document-view-related commands. CommandEvents include the user ID attribute so it is possible to identify the originator of the command. This is useful for group undo/redo [18] as well as for group activity awareness and for controlling access rights.

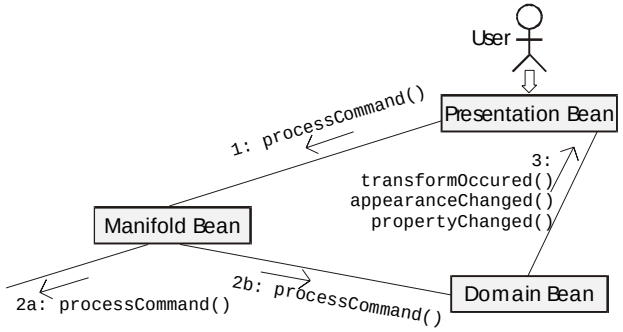

Figure 3: Event exchange and interception in a Manifold bean. Commands generated by the local user that affect the domain bean. Action 2a passes the command to the collaboration bus which broadcasts it to the remote peers.

The domain and the presentation beans need to be glued together since the execution environments assume single beans. The third bean, the Manifold Bean, interacts with the collaboration bus to send and receive collaboration events. This interaction is through the Java Beans' event mechanism: DISCIPLE subscribes for the bean's events and replicates them to the collaborating peers. Figure 3 shows the interactions between the beans. A local command is dispatched to the local domain bean and the collaboration bus either simultaneously or first to the bus and then locally after the command is reflected back. Which strategy is used depends on the employed concurrency control algorithm, optimistic or pessimistic.

The architecture of the Manifold framework is lightweight, scalable, and extensible. Dewan [3] argues that any collaborative application can be seen as a generalized editor of semantic objects defined by it. A user interacts with the application by editing a rendering of these objects using text/graphics/multimedia editing commands. In addition to passive editing, in a generalized editor, the changes may trigger computations or behaviors in the objects. In that sense, Manifold can be used to build arbitrary groupware applications that deal with structured documents. The example applications given below use Manifold for graphics editors with 2D vs. 3D representations of objects.

XML AND MANIFOLD BEANS

So far we have presented an architecture which can be used to build heterogeneous groupware. However, once built, the beans do not allow for dynamic customization and thus, adaptation to the current context. Since Java supports dynamic loading of individual classes, we could exploit this feature and consider the above classes as components in a repository, which can be selected at run-time so to customize the application's document, the way it is visualized, and the way the entire application is generated. We have chosen XML as the language for such purpose.

The XML programmer creates XML and XSL files based on, respectively, the sets of available Glyphs and GlyphViews as well as their characteristics. Manifold Glyphs correspond to elements in the source tree and

GlyphViews correspond to elements in the result tree. Attributes in the result tree are mapped to the Glyph properties. For example, if the *result node* corresponding to a Glyph has an attribute `COLOR` with a value of `BLUE`, the Glyph's property "color" is initialized in blue. However, the correspondence between the elements and Glyphs is not one-to-one. Not all XML elements are Glyphs. For example, some elements could describe Glyph properties. A 3D transformation could be represented as:

```
<TRANSFORMATION>
… 4x4 matrix defining the 3D transformation …
</TRANSFORMATION>
```

This element could even have sub-elements if the transformation is represented via the axis-angle, scale and translation parameters, each tagged individually.

The result tree is used to create the domain scene graph. A key benefit of implementing presentation and domain as distinct beans rather than the whole package as a single bean is in being able to mix and match different combinations. We can have a set of more or less complex beans for each. Different domain beans can implement complex behaviors and the presentation beans can implement visualizations with varying realism. Depending on the context, different domain/presentation pairs can be downloaded on demand. In addition, each bean in the pair may run on different machines. The presentation bean may run on a hand-held device, and the domain bean would run on a PC embedded in the environment (base station).

In heterogeneous applications, the scene graphs are populated with different Glyphs having different properties. The Glyph operations do not directly apply to remote Glyphs. We need to transform the operations to apply them on remote sites (Figure 4). In the worst case, when all sites have different domains, there are $n*(n-1)$ mappings. The agency that does the mappings can be a central server or, in an entirely distributed case, each client can perform the required mappings on its own. This latter approach is not likely to be practical for lightweight terminals, so proxy agencies are needed to perform the mappings.

For instance, in the example applications given below, the event communication is implemented as follows. In order not to have the lower-end machines do the mapping, multiple communication channels are used. The low-end beans (Flatscape) only communicate on their dedicated channel, while the high-end beans (cWorld) communicate on both the Flatscape channel and the cWorld channel (Figure 5). The mapping and translation between the two domains is therefore only performed on higher-end machines (running cWorld).

Another issue deals with the actual implementation of the mappings. In the above-mentioned example applications, the main difference in Glyphs is the use of 2D vs. 3D representations for the same objects. In particular, the transformation matrices that describe position and orientation of the objects are very different. The mapping

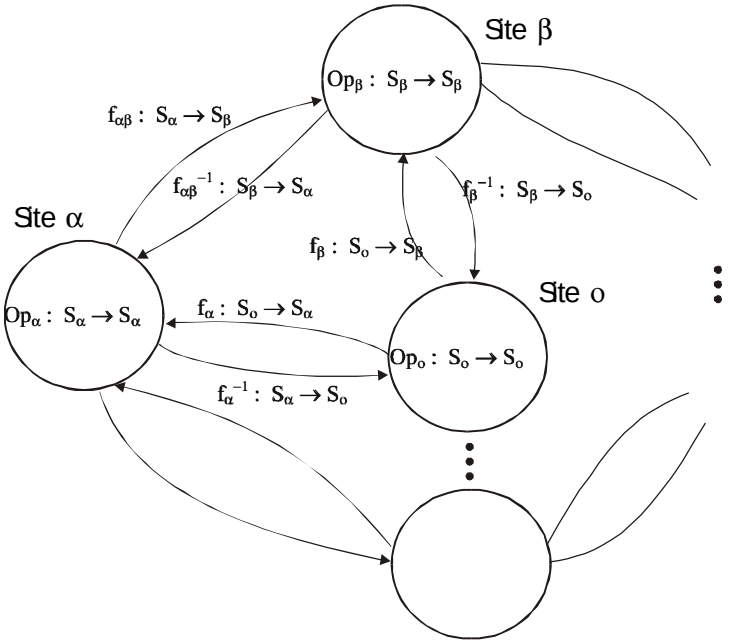

Figure 4: Operations and mappings between the domain scene graphs (*S*) in a heterogeneous collaborative system. A central site o could maintain a common model to simplify the mappings to/from different sites.

between the different domains is defined using XML documents, so the end user can easily redefine and customize the mappings. In order to improve the performance of the system, the XML document is used to instantiate a look-up table at the beginning of the session. The table is used for the actual mapping at run-time.

EXAMPLE APPLICATIONS

Using the Manifold framework we developed two complex applications: a 2D graphics editor (Flatscape) and a 3D virtual world (cWorld). Both applications currently run on desktop workstations. The heterogeneity is simulated both by using different display applications and by restricting the window size of the Flatscape environment to the screen size of a typical Windows CE handheld computer (320×240 pixels).

cWorld

The cWorld Java Bean (Figure 6) enables synchronous, multi-user building of collaborative virtual environments. It uses Java3D. CWorld does not require special hardware and can be operated using the keyboard and a mouse although it also supports the use of the Magellan SPACE Mouse. This device provides the six-degrees of freedom

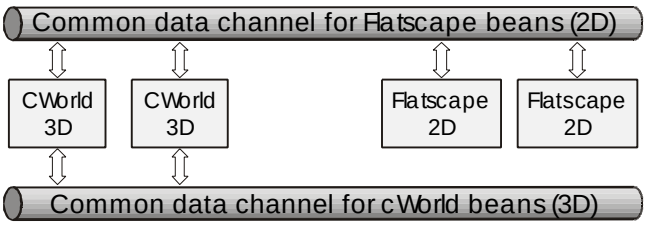

Figure 5: Data communication among Flatscape and cWorld beans.

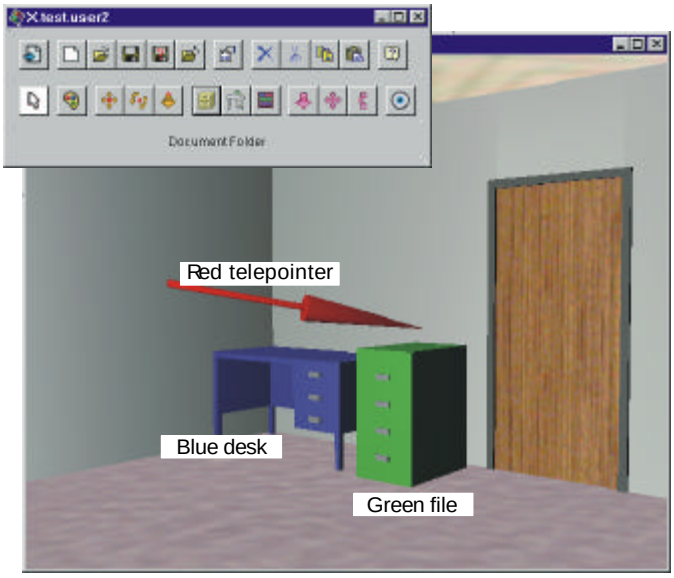

Figure 6: A sample CVE built using cWorld. The figure also shows a three-dimensional telepointer.

movement used in navigating 3D spaces.

The operations on the Glyph tree in cWorld are:

Op_1: creates simple geometric figures (box, cylinder, cone, sphere), lights (directional, point, spotlight), and furniture objects (desk, cabinet, bookcase), which are PolyGlyph objects. The room object is a PolyGlyph as is the telepointer (cylinder + cone).

Op_2: deletes any of the above listed Glyphs

Op_3: sets the Glyph properties, such as color, texture, position, dimensions, and manipulation constraints (e.g., furniture objects are constrained from 'flying'). The lights have additional properties, such as direction, attenuation, spread angle, etc. The user can also apply 3D affine transformations to Glyphs.

Each user has a unique 3D telepointer, shown in Figure 6. These devices function as primitive avatars and appear at the discretion of the user. The telepointer is drawn at the position and orientation of the user's line of sight.

Flatscape

The application is developed using Java2D. An example snapshot is shown in Figure 7.

The operations on the Glyph tree in Flatscape are:

Op_1: creates simple geometric figures (line, rectangle, ellipse, polygonal line, spline) and bitmap images. The figures can be grouped into PolyGlyphs, like in the case of the room object.

Op_2: deletes any of the above listed Glyphs

Op_3: sets Glyph properties, such as contour color/thickness/dash, fill color, position, dimensions, visibility, and manipulation constraints (e.g., movement only along the *x*-axis). The lines also have arrow style and the splines have other properties. The user can also apply 2D affine transformations to Glyphs.

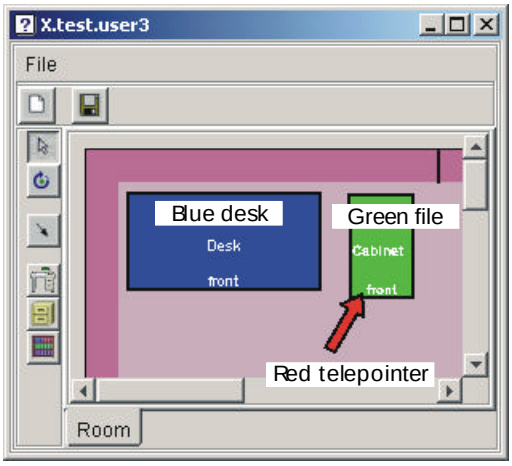

Figure 7: A room floor plan as seen using Flatscape, corresponding to the room shown in Figure 6. The figure shows also a two-dimensional telepointer.

The telepointers in Flatscape are 2D arrows pointing away from the center of the room towards the part of the room currently viewed by the user. This is different from the typical telepointer, e.g., that found in [19], which only shows position and not orientation. In our 2D implementation the telepointer owner, rather than the recipient, has control over its visibility.

EVALUATION

Flatscape and cWorld demonstrate that our framework supports heterogeneous collaborative environments. Although we show that we can construct these environments, we do not know how effective they will be for collaboration. It is readily assumed that a common view is necessary for effective communication, but others have found that non-WYSIWIS systems do not impair collaboration and may even facilitate it [21]. Obviously this depends on how different the shared views are. Our 2D vs. 3D displays represent a common future difference expected in office-field communication. We evaluate the effect of this difference in a short study. In a 3D to 2D communication, we expect users to know the differences between their view of the information and that of their collaborators. However, we do not know if the users will readily adjust themselves to this difference in the actual task. The cognitive load of carrying out the collaboration may be too high.

Each worldview gives its user useful information. Since the 2D environment is designed to simulate a PDA, only a small part of the scene is visible, and the user needs to scroll in order to gain an entire representation of the office space. Jones *et al.* [10] show that this scrolling increases task performance time. The 2D space also leaves out useful information such as the orientation (front vs. back) of the objects and the aesthetic placement of the objects next to each other. In the 3D view, aligning objects with walls and corners of rooms is more difficult because distance and alignment are harder to estimate. On the other hand, 3D

users have more cues available for correct orientation and aesthetic placement of objects.

Navigation in the 2D space is done by left/right and up/down scrolling through the top down view of the world. In contrast, the 3D user navigates with 6 degrees of freedom controlled by Magellan SPACE mouse movements. 3D users have no external frame of reference (top, bottom, left, right) and must communicate about their position and viewpoint using common features in the space. The user also sees the space from different locations in the room so that there is no sense of a left wall or a front wall.

Each view of the environment therefore gives users certain advantages for creating and placing objects on their display. However, their different views of the world are likely to lead to misunderstandings in the communication that transpires. We expect these misunderstandings to be recognized (because of the observed changes on each user's display) and immediately repaired through the verbal channel. These repairs are likely to be quick and considered a natural part of the collaboration. However, they are also likely to increase the amount of time required to perform the collaboration task. We examine whether this is true in a study that compares collaboration across the 2D–3D heterogeneous environment to collaboration in both the 2D–2D and 3D–3D homogenous environments.

Overview of Study

In the H vs. Ht (Homogenous vs. Heterogeneous) study, we collected four types of data:

1. The number of communication repairs that occurred in a shared room arrangement task.
2. The amount of time needed by a collaborative team to perform each room object arrangement.
3. The accurate placement of the objects in the task.
4. The perception that users had about the quality of the communication in the H vs. Ht environments.

The study was a within-teams design, that is, collaborative teams were assigned to each of four conditions, 2D–2D communication, two 2D–3D communication arrangements and 3D–3D communication. We expected the heterogeneous collaborations to experience more communication repair statements, and to take somewhat more time than the homogenous collaborations. This is because communication repair is expected to increase the overall time for tasks. We expected fewer repairs in the homogenous tasks because synchronized views give the collaborators equivalent information. User perception is not expected to be different for the two communication environments, in part, because we assessed this perception at the end of all trials not when communication difficulties were actually occurring and, in part, because we did not expect users to be aware of communication difficulties unless they could not be fixed by verbal repair.

We were most interested in the differences in time and communication repair that occurred. If they were small and went unrecognized by the users, we then felt that implementation of heterogeneous environments in office-field situations was viable.

Subjects

We selected 24 subjects (6 females and 18 males) from the university undergraduate and graduate population. All of the subjects had high levels of experience with computers and most engaged in video games. Only four subjects had never played video games, seven had been playing for 14 or more years and the rest had spent at least 1 year playing video games. Twenty-one subjects still played video games between one and five hours per week. Only one subject reported playing video games for more than five hours per week. All participants indicated they were comfortable using a computer and mouse and eight had previous experience with 3D virtual environments. We assigned our subjects to groups of three. Five of the teams were made up of members who knew each other.

Procedure

The study required each team to perform four office furniture arrangement tasks. In each task, a total of 9 pieces of furniture (bookcases, file cabinets and desks) had to be placed at specified positions in the room. Each team member was given a sheet of paper with a top-down view of the placement of 3 of the objects. They could not, however, place these objects themselves and needed to direct other team members in their placement. Color indicated the mover of the objects and the owner of each telepointer. For each of the tasks, we assigned a display configuration for each of the team members so that they were either in a 2D or 3D environment. The assignment of the display environments is shown in Table 1.

Table 1: Assignment of display environments to subjects.

	User1 (red)	User2 (green)	User3 (blue)
Task 1	2D	2D	2D
Task 2	2D	2D	3D
Task 3	3D	3D	2D
Task 4	3D	3D	3D

We used eight 3-member teams. Team members were placed in different locations in a large lab and connected by microphones and headsets. Participants used Windows NT workstations connected to Ethernet. Workstations were equipped with both a normal PC mouse and a Magellan SPACE mouse. Before the study, subjects had 10-15 minutes training with cWorld using the Magellan SPACE mouse, and 5-10 minutes training with Flatscape.

Each 2D user had a Flatscape bean to accomplish the collaborative task. Flatscape presented only the top view of the room and users needed to use vertical and horizontal scrollbars to navigate through the room.

Each 3D user had a cWorld bean to accomplish the collaborative task. Navigation and manipulation of objects were performed using the Magellan SPACE mouse. User navigation in the cWorld was restricted to

forward/backward movement on the floor as well as yaw and pitch rotations of the viewpoint. Object navigation was restricted to horizontal displacements on the floor and rotation around a vertical axis.

Color differentiated among the three users' objects and telepointers. Red was assigned to user1, green to user2 and blue to user3. In Figures 6 and 7, user1 is showing his telepointer, user2 has just added a file cabinet and user3 has placed a desk in the scene.

In each task users were provided with the partial layouts and instructed to collaboratively furnish the room. Each user used both Flatscape and cWorld twice. Figure 8 illustrates a subject performing one of our office arrangement tasks.

We collected data in our study via three observers. Each observer had a large digital display clock that they used to record the start and stop time for each object placement request along with the environment of the collaborators (2D or 3D). During the communication about the object placement, the observers counted the number of communication repairs that were required, e.g., "no, the desk is facing the wall, rotate it 180 degrees so it faces outward into the room." We also distinguished between whether a user was requesting an action or performing an action in the Flatscape-to-cWorld exchange. Three types of repairs were counted: a request to adjust a wrongly placed object, a request to re-orient an object, and an indication that the wrong object was being moved. Table 2 gives an example of the type of data collected.

Table 2: Example of data collected in H-vs.-Ht study.

Object type	Start time	End time	No. of repairs	Request from	Perform by
Red desk	9:04:32	9:06:20	4	2D	3D
Green file	9:12:02	9:13:21	1	3D	2D

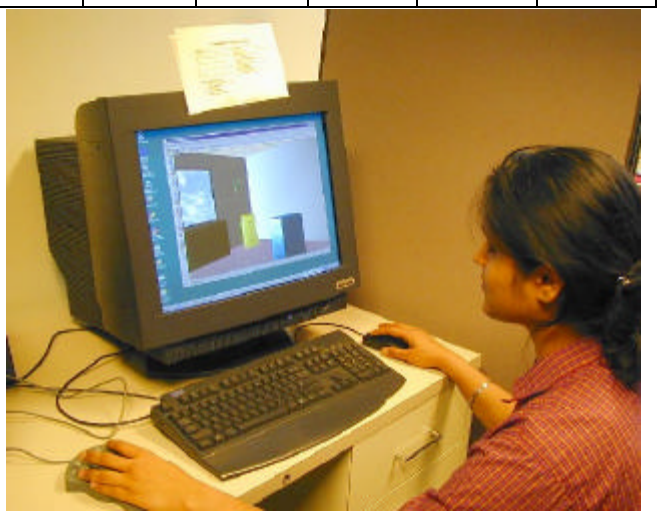

Figure 8: A participant in a collaborative session. Note: the paper posted at the top of the screen lists navigation and object manipulation hints.

Results

We did not find the expected differences between heterogeneous and homogenous environments. Table 3 presents the descriptive statistics of our measures.

Table 3: Average time and number of communication repairs per object placement for 2D–2D, heterogeneous and 3D–3D collaboration. The time is presented in seconds.

	2D–2D	2D–3D	3D–2D	3D–3D
Mean time / std. dev.	79 / 119	121 / 99	79 / 128	110 / 111
Mean no. of repairs / std. dev.	2.02 / 1.93	3.53 / 2.71	1.26 / 1.15	1.91 / 2.71

As can be seen from Table 3, we found no large differences in either the task performance times or the mean number of repairs. The largest differences are in the 2D-to-3D collaborations, where the individual viewing a Flatscape display is giving object placement instructions to a person in cWorld. A correlation of the object placement times and the repair counts (Pearson $r = 0.16$, n.s.) indicated that there was little relationship between the number of repair statements and the time it took to perform the task.

Because of this low correlation, we felt comfortable in separating out the number of communication repairs and the object placement times in our multivariate data collection and running a Student *t*-test that compared heterogeneous to homogenous collaboration environments. Our results are shown in Table 4.

Table 4: A comparison of communication repair and task performance time for H-vs.-Ht collaboration environments.

	Heterogeneous 2D-3D & 3D-2D	Homogenous 3D & 2D	Student *t* test results
Mean time / std. dev.	99 / 116	94 / 116	$t = 0.337$, $p = 0.37$
Mean # repairs / std. dev.	2.37 / 2.35	1.97 / 2.34	$t = 1.337$, $p = 0.18$
No. of objects	188	92	

The two *t*-tests show no significant differences between the heterogeneous and homogenous conditions.

We also examined four key questions on our post study questionnaire. The questions assessed how subjects felt about the H-vs.-Ht collaboration environment. Subjects responded on a scale from 1 (strongly agree) to 4 (strongly disagree) to the following four questions:

Q1. While using 2D, I found it difficult to collaborate with users using 3D applications.
Q2. While using 3D, I found it difficult to collaborate with users using 2D applications.
Q3. While using 2D, I found it difficult to collaborate with users using 2D applications.

Q4. While using 3D, I found it difficult to collaborate with users using 3D applications.

The average results from these questions are showing below in Table 5.

Table 5: User perceptions of collaboration in each of the two collaboration environments.

	Average Score (from 1 – strongly agree to 4 – strongly disagree)
Q1 (2D–3D)	2.9
Q2 (3D–2D)	2.8
Q3 (2D–2D)	3.3
Q4 (3D–3D)	2.9

Overall, the users did not find the collaborations difficult, but they found the 2D–2D collaboration to be easier than anything involving 3D collaboration.

Discussion

We expected the homogenous collaboration to show better performance times and less repairs than the heterogeneous collaboration. However, we were not able to find these differences. We believe that this is because of two additional factors; (1) the difficulty of placing objects in the 3D environment and (2) the advantage of viewing objects in the 3D environment. The object placement times in the 2D–2D collaboration (Table 3) were, on average, 31 seconds faster than the 3D–3D collaboration. The mean number of repairs for these tasks was approximately equal. The implication is that the object alignment task in 3D is more difficult (our users mentioned this numerous times).

We examine this explanation further by looking at the 2D–3D and 3D–2D tasks. In the 2D–3D task, the person giving directions was in the 2D environment and the person performing the object placement was in the 3D environment. This was reversed in the 3D–2D task. If the difficulty were with the 3D environment, we should find that the average object placement time is longer in the 2D–3D case. The average time difference was significantly longer ($t = 5.149$, $p < 0.001$). Table 6 displays the results of our post hoc comparisons.

Table 6. A comparison of the communication repair and task performance time between 3D–2D and 2D–3D object placement requests.

	3D-2D	2D-3D	Student *t* test results
Mean time / std. dev.	79 / 128	121 / 99	$t = 5.149$, $p < 0.001$
Mean # repairs / std. dev.	1.26 / 1.15	3.53 / 2.71	$t = 1.746$, $p = 0.086$
No. of objects	47	45	

We also note that the mean number of repairs is considerably less for the 3D–2D than the 2D–3D object placement. This result is tending towards significance ($t = 1.746$, $p = 0.086$). We interpret this effect to arise for two reasons. The 3D view allows the team member to give better directions to the 2D member and the 2D application is easier to perform manipulations in. This suggests that heterogeneous collaboration may actually be better than homogeneous collaboration as long as it is 3D–2D. This is similar to work carried out by Steed *et al.* [20] who found that leaders emerged in a group task from the more immersive VR environment.

The results from our post-test questionnaire also show that users found the 2D environment to be easier to use. They found collaboration to be easiest in the 2D environment even though our results show that collaboration was best in the 3D–2D environment.

We also observed throughout the study that users readily adapted to the differences in their environments. For example, 2D users soon realized that phrases such as "left wall" made little sense to their 3D counterpart and adjusted their conversation accordingly. Furthermore, most teams did realize that it is easier to correctly align objects in 2D, and therefore asked a 2D user to perform the alignment. There were also occasions of transference between the 2D and 3D tasks. One user, for example, looked for other users' telepointers on the floor when first entering the 3D environment.

At the time of this paper generation, we have not yet teased out the effect of differences in experience with 2D and 3D environments nor have we evaluated the placement accuracy of the collaborative tasks. We recognize that learning went on in the four trials and that we did not randomize the order of the trials across the teams. This was done because subjects came to the study possessing mouse skills and not 3D skills. Placing the 3D–3D study last gave subjects more time to be familiar with the 3D movements and orientation. Since we were running the experiment to capture differences between the heterogeneous and homogenous environments, this order worked in opposition to the results we were seeking.

CONCLUSION

The need for heterogeneous sharing in synchronous collaboration grows ever stronger with the proliferation of diverse computing environments, particularly wearable and handheld computers. This work presents a data-centric design for synchronous collaboration of users with heterogeneous computing platforms. It is a significant departure from the traditional model of sharing, which is application- or procedure-centric. Our approach allows clients with different capabilities to share different subsets of data in order to conserve communication bandwidth. XML, which serves as the communication medium, is a standard that has already gained great acceptance, and provides a powerful medium for both data exchange and visualization specification.

We also illustrate via an extreme example of size and dimensionality differences that heterogeneous collaboration does not appreciably affect task performance and that users

perceive the task performance to be equivalent to homogenous environment collaboration.

Further information about the DISCIPLE project and heterogeneous collaboration is available at:

http://www.caip.rutgers.edu/disciple/

ACKNOWLEDGMENTS

D. Makwana, A. Francu, and K. R. Pericherla contributed significantly to the software implementation. The research reported here is supported in part by NSF KDI Contract No. IIS-98-72995 and by the Rutgers CAIP Center.

REFERENCES

1. Billinghurst, M., Bee, S., Bowskill, J., and Kato, J. Asymmetries in collaborative wearable interfaces. *Proc. 3rd Int'l Symposium on Wearable Computers*, San Francisco, CA, pp.133-140, October 1999.
2. Calder, P.R., and Linton, M. A. Glyphs: Flyweight objects for user interfaces. *Proc. 3rd ACM Symposium on User Interface Software and Technology (UIST)*, Snowbird, UT, pp.92-101, October 1990.
3. Dewan, P. Architectures for collaborative applications. In *Computer Supported Co-operative Work*, M. Beaudouin-Lafon (Ed.), John Wiley & Sons, Chichester, England, pp.169-193, 1999.
4. Dewan, P., and Sharma, A. An experiment in interoperating heterogeneous collaborative systems. *Proc. 6th European Conf. on Computer Supported Cooperative Work (ECSCW'99)*, Kluwer Acad. Publ., Copenhagen, Denmark, pp.371-390, September 1999.
5. Engelbart, D. Authorship provisions in AUGMENT. *Proc. IEEE Compcon Conference*, 1984.
6. Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley Longman, Inc., Reading, MA, 1995.
7. Garfinkel, D., Welti, B. C., and Yip, T. W. HP SharedX: A tool for real-time collaboration. *Hewlett-Packard Journal*, 45(2):23-36, April 1994. http://www.hp.com/hpj/94apr/toc-04-9.htm
8. Higgins, M., Lucas, P., and Senn, J. VisageWeb: Visualizing WWW data in Visage. *Proc. IEEE Symp. Information Visualization (InfoVis'99)*, San Francisco, CA, October 1999.
9. Hindmarsh, J., Fraser, M., Heath, C., Benford, S., and Greenhalagh, C. Fragmented interaction: Establishing mutual orientation in virtual environments. *Proc. ACM Conference on Computer-Supported Cooperative Work (CSCW'98)*, Seattle, WA, pp.217-226, Nov. 1998.
10. Jones, M., Marsden, G., Mohd-Nasir, N., Boone, K., and Buchanan, G. Improving Web interaction on small displays. *Proc. 8th Int'l World Wide Web Conference*, Toronto, Canada, May 1999.
11. Kahneman, D. *Attention and Effort*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973.
12. Karsenty, A., Tronche, C., and Beaudouin-Lafon, M. GroupDesign: Shared editing in a heterogeneous environment. *USENIX Computing Systems*, 6(2),pp 167-195, Spring 1993.
13. Kraut, R., Miller, M. D., and Siegel, J. Collaboration in performance of physical tasks: Effects on outcomes and communication. *Proc. ACM Conference on Computer-Supported Cooperative Work (CSCW'96)*, Boston, MA, pp.57-66, November 1996.
14. Maya Design Group, Inc. Visage Link. http://www.maya.com/visage/base/technical.html
15. Myers, B.A., Stiel, H., and Gargiulo, R. Collaboration using multiple PDAs connected to a PC. *Proc. ACM Conf. on Computer-Supported Cooperative Work (CSCW'98)*, Seattle, WA, pp.285-294, Nov. 1998.
16. Olsen, D. R., Hudson, S. E., Phelps, M., Heiner, J., and Verratti, T. Ubiquitous collaboration via surface representations. *Proc. ACM Conference on Computer-Supported Cooperative Work (CSCW'98)*, Seattle, WA, pp.129-138, November 1998.
17. Patterson, J. F., Hill, R. D., Rohall, S. L., and Meeks, W. S. Rendezvous: An architecture for synchronous multi-user applications. *Proc. ACM Conference on Computer-Supported Cooperative Work (CSCW'90)*, Los Angeles, CA, pp.317-328, October 1990.
18. Prakash, A. Group editors. In *Computer Supported Co-operative Work*, M. Beaudouin-Lafon (Ed.), John Wiley & Sons, Chichester, England, pp.103-133, 1999.
19. Roseman, M., and Greenberg, S. Building real-time groupware with GroupKit, a groupware toolkit. *ACM Transactions on Computer-Human Interaction*, 3(1): 66-106, March 1996.
20. Steed, A., Slater, M., Sadagic, A., Bullock, A., and Tromp, J. Leadership and collaboration in shared virtual environments. *Proc. of the IEEE Virtual Reality Conference*, Houston, TX, pp.112-115, March 1999.
21. Stefik, M., Bobrow, D. G., Lanning, S., and Tatar, D. WYSIWIS revised: Early experiences with multiuser interfaces. *ACM Transactions on Information Systems*, 5(2):147-167, April 1987.
22. Tatar, D. G., Foster, G., and Bobrow, D. G. Design for conversation: Lessons from Cognoter. *International J. of Man-Machine Studies*, 34(2):185-209, 1991.
23. Wang, W., Dorohonceanu, B., and Marsic, I. Design of the DISCIPLE synchronous collaboration framework. *Proc. 3rd IASTED Int'l Conference on Internet, Multimedia Systems and Applications*, Nassau, The Bahamas, pp.316-324, October 1999.
24. Wireless Application Protocol Forum. Wireless markup language specification SPEC-WML-19990616.pdf. At: http://www.wapforum.org/what/technical.htm