

# Tailorable Domain Objects as Meeting Tools for an Electronic Whiteboard

Thomas P. Moran, William van Melle, and Patrick Chiu\*

Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304

{moran,vanmelle}@parc.xerox.com, chiu@pal.xerox.com

## ABSTRACT

Our goal is to provide tools to support working meetings on an electronic whiteboard, called Tivoli. This paper describes how we have integrated structured “domain objects” into the whiteboard environment. Domain objects represent the subject matter of meetings and can be exchanged between Tivoli and group databases. Domain objects can be tailored to produce meeting tools that are finely tuned to meeting practices. We describe the facility for tailoring and managing domain objects and the user interface techniques for blending these into the whiteboard environment. We show examples of both specific and generic meeting tools crafted from domain objects, and we describe a long-term case study in which these tools support an ongoing work process.

**Keywords:** whiteboard metaphor, pen-based systems, freeform interaction, implicit structure, informal systems, recognition-based systems, list structures, meeting support tools, gestural interfaces, user interface design, tailorability, customization, object-oriented user interfaces

## 1. INTRODUCTION

Our goal is to provide computational support for working meetings, in which small groups of people collaborate in real time. By “working meeting” we mean group interactions that work with a large set of knowledge material (assessing, discussing, organizing, negotiating about the materials, creating new materials, etc.). Group interaction in such situations is freewheeling and fluid. Further, such meetings are usually part of a larger work process in which there is a knowledge base of materials that is used between meetings. During a given meeting, only a relevant “working set” of materials is needed to effectively conduct the meeting.

Groups often use whiteboards in meetings to provide a shared visual surface to represent the materials they are

working with and to preserve a shared context. Pen-based computational systems that allow scribbling and gesturing on wall-size displays can support a whiteboard metaphor for working meetings.

We have been working for several years on a program of research to provide computational meeting tools based on a whiteboard metaphor. Our idea is to allow freeform creation and manipulation of materials on the whiteboard and to provide facilities to easily organize and structure the materials as needed by the group.

Our program of research is based on the LiveBoard [3], a large, shared, pen-based electronic (rear-projected) display. This provides a whiteboard-size interactive display that allows us to experience and experiment with “group-computer interaction” in a way not possible with smaller displays, such as workstations. We have developed a software application, called *Tivoli* [16], that simulates whiteboard functionality on the LiveBoard (or any other display). Tivoli provides basic pen-based scribbling and editing with pen-based gesturing and wiping techniques. In this paper, we use the term “board” to refer to an interactive electronic whiteboard.

We extended Tivoli [12] to allow “implicit structuring” of the material on the board. Material is created on the board in a freeform manner, which means that anything can go anywhere without constraint. However, the user can indicate by certain gestures that the material is to be regarded as temporarily structured in some manner, e.g., as handwritten text. Tivoli will then apply editing operations in accordance with the conventions of that structure, such as opening and closing space where needed when moving words around.

We then extended Tivoli by developing a set of simple and natural techniques for helping users spatially organize material on the board [11]. The most important is to provide ways to group materials into regions on the board. We provide for both rectangular regions and for freeform en-

*Proceedings of CSCW'98.*

---

\*Current Address: Patrick Chiu, FX Palo Alto Laboratory, 3400 Hillview Avenue, Palo Alto, CA 94304. At the time of the work reported in this paper, Patrick Chiu was with Liveworks, Inc., A Xerox Company, working at Xerox PARC.

closures of any shape. Tivoli can thus limit structured operations to the regions in which they are invoked.

These gestural pen-based techniques result in a fluid, graceful user interface that integrates the informality and flexibility of the whiteboard metaphor with the power of a graphical editing system.

In this paper, we build on the user interface and structuring features of Tivoli by adding a facility for handling new kinds of objects, *domain objects*, in Tivoli. Before this, all of the materials on the Tivoli board have merely been graphical objects, whose interpretation or relationship to other data exists only in the minds of the users. For example, suppose a group's current action items were exported from a workflow system as a prioritized list and read (as simple text) onto the Tivoli board; then users in a meeting edited, erased, added, and rearranged the action items on the board. The resulting board image would bear no relationship to the action item entities in the workflow system. However, by representing the action items as domain objects, these entities can be imported into Tivoli, manipulated *in a gestural whiteboard style* during a meeting, and then the results (annotated and rearranged priorities) can be exported back to the workflow system. Thus, domain objects represent the subject matter (i.e., domain of knowledge) of meetings. We use domain objects as the medium of exchange with repositories of work knowledge. We are able to tailor specialized domain objects as meeting tools that are tuned to particular contents and processes of meetings.

This paper proceeds by first focusing on our own work, motivating and describing our design of the domain object facility. We present an example of a working meeting that motivates our approach (Section 2), which we generalize to a set of objectives for the facility (Section 3). We contrast our approach with others (Section 4) and then describe the domain object facilities in detail (Section 5). We show how domain objects can be tailored to support different types of meetings (Section 6). Then we show how we have incorporated domain objects into a longitudinal case study of a real

work process (Section 7). Finally, we step back and discuss several issues of our approach and compare it to other work (Section 8).

## 2. EXAMPLE OF A WORKING MEETING

Consider an actual working meeting at PARC (in May 1995). It was the fifth in a series of meetings. In the first four, different groups brainstormed a set of 34 scenarios for a system being designed. The goal of the fifth meeting was to select a few scenarios for further study. In the previous meetings, scenario descriptions were captured on Excel spreadsheets, which were consolidated into a single list. In the fifth meeting, however, the group quickly abandoned the spreadsheet, since it was too difficult to see the overall picture and too awkward to manipulate.

Instead the list of scenario names was imported into Tivoli, in a four-column format, so that all the scenario names could be seen at once (see Figure 1a). The group spent 50 minutes discussing the various scenarios, moving the names around to group them into categories by similarity, and then selected six of them for further work. The result of this work (Figure 1b) is a complete rearrangement of the items, plus scribbled annotations.

The group faced a challenging task, and the method of accomplishing it was not understood at the beginning of the meeting. Once they started moving items in Tivoli, the strategy and process came naturally, which involved a lot of physical manipulation (over 50 item moves). Several critical features of the tools made this meeting effective:

- A representation of previous work was needed as a starting point.
- The materials needed to be easily and freely manipulated on the 2D surface of the board in order to discover a satisfactory categorization.
- Handwritten annotations were intermixed with the initial textual materials.
- The items and annotations were spatially arranged into regions of similar items. The circled numbers were used to indicate when clustered items made sense.

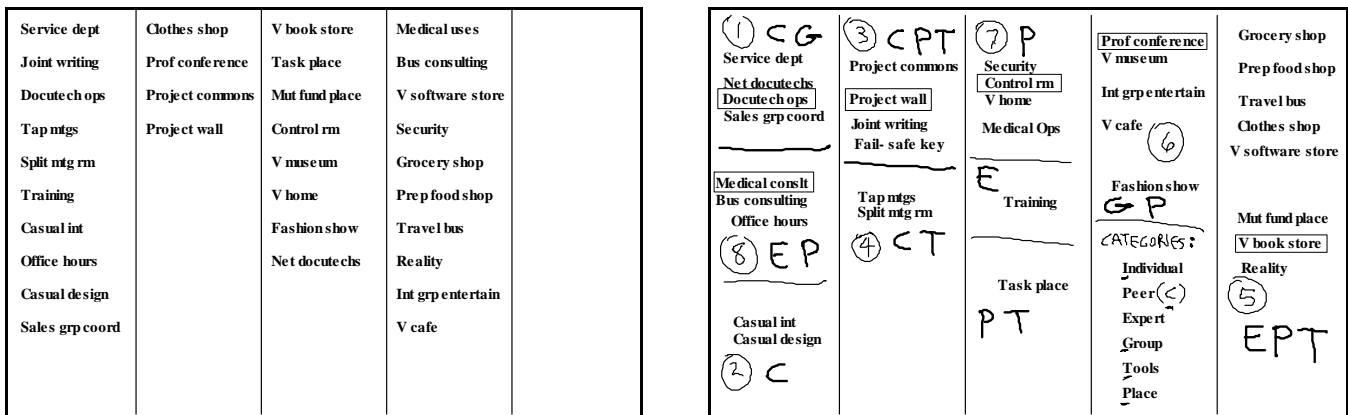


Figure 1. Tivoli Board for the Scenarios Meeting: (a) initial state; (b) final state.

- Category annotations were used to help articulate the similarities and differences among the groups.
- Representative items were selected by coloring them red (shown as boxed items in Figure 1b), and the overall pattern of the selections was assessed.

Note that the materials on the board were treated as a set of objects to be arranged and annotated and that this was the medium for carrying out the work that had to be done.

### 3. OBJECTIVES FOR MEETING TOOLS

From this example and other experiences with Tivoli in different meetings, we formulated objectives for the design of a facility for meeting tools on an electronic whiteboard:

1. The system must be able to represent the *semantic content* of meetings, the things meetings are about. We encoded these “things” as *domain objects*, i.e., objects that represent meaningful entities in the subject matter domain that end users think about and consciously work with.
2. The domain objects must be *connected with the “workbase”* (e.g., database, document management repository, workflow system) used outside of meetings.
3. The domain objects must be incorporated into a *whiteboard style of interaction* (rapid, easy, pen-based and gesture-based) to support, and not inhibit, the way meetings work.
4. The whiteboard representation must include informal scribbled *annotations* as well as domain objects, since meetings never stay within the confines of a formal representation. The main user interface issue is to effectively accommodate the *blending of structured and freeform* materials on the board.
5. The system must be able to *interpret the 2D spatial arrangement* of the domain objects on the board, for the spatial arrangement is the visible shared representation meeting participants use to develop a shared understanding.
6. The format and behavior of the domain objects must be *tailorable* to the special needs of different meetings. Meetings are not isolated events, but are part of a larger work context. While there are generic aspects to meetings, most efficient meetings are characterized by adopting practices that are tuned to the work to be done in the particular work context.

### 4. APPROACHES

There are several different approaches to the design of a meeting tools facility:

1. Use standard *desktop-style software tools*. While users are familiar with these tools, annotations are difficult to do. The tools are rigid and usually not adaptable to unexpected turns that take place in meetings.

2. Import *snapshots* from standard tools under a freeform drawing layer, such as can be done with MeetingBoard.<sup>1</sup> This makes annotations easy, but the contents of the snapshots are not manipulable.
3. *Embed* the desktop applications within a freeform whiteboard environment using an object embedding technology such as OLE or ActiveX [2]. The workbase objects are manipulable and editable within the embedded application, and annotations can be made around it. However, this can result in a confusing mix of user interfaces (different UIs for different applications); and the whiteboard’s freeform manipulation techniques cannot be used within the embedded applications.
4. Use *domain objects* in a freeform whiteboard environment. The difference from the previous approach is that semantic data is packaged into objects in the whiteboard application. Thus the domain objects are treated as whiteboard graphic objects, and the whiteboard user interface is extended by tailoring the domain objects to have specialized behaviors appropriate to their use.

Note that in this paper we use the unmodified term “object(s)” to refer to domain object(s).

### 5. THE DOMAIN OBJECT APPROACH

We have implemented domain objects as tailorable graphic objects in Tivoli. Domain objects are significant at two levels: (1) as a part of an architecture that allows Tivoli to participate in a larger collaborative support system and (2) as new user interface techniques in the Tivoli whiteboard environment.

#### Architecture of Domain Objects

The architecture is illustrated in Figure 2. We assume some kind of workbase (e.g., Lotus Notes) to persistently store the materials of a collaborative work process. People can interact with this workbase with various client applications on their desktop workstations, thus providing asynchronous coordination of individual work through the workbase. However, desktop applications are not effective in meeting settings; Tivoli is a more appropriate user interface. While we can put and get Tivoli files (containing editable page images) from the workbase, these files are a separate representation from the other data in the workbase. The issue is how to use the Tivoli user interface with this other data.

The strategy we follow is to provide a script to import data from the workbase and translate them into a set of domain objects and page layouts in a markup file, which can be read into Tivoli. The meeting participants interact with Tivoli, manipulating the objects with the whiteboard user interface. The altered objects are then exported and trans-

---

<sup>1</sup> MeetingBoard (LiveWorks, Inc.) is the whiteboard product based on Tivoli.

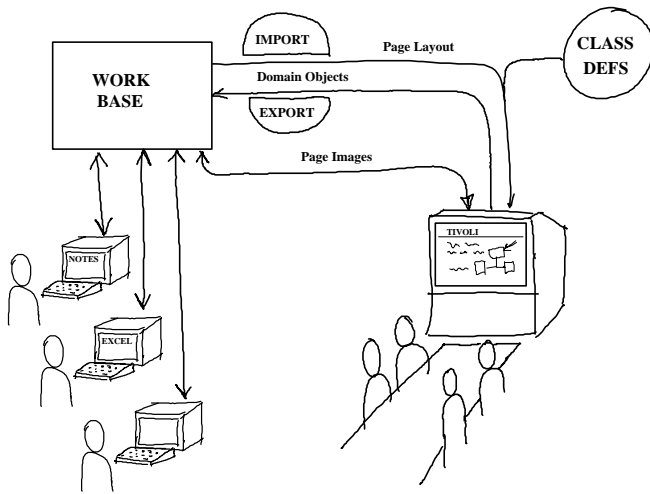


Figure 2: Architecture of Domain Objects

lated back to the workbase representation. Thus, the results of meeting activities are fully integrated into the workbase.

Tivoli has facilities for representing, defining, and managing domain objects. A domain object is a cluster of data representing a work entity (a document, an action, a person, an idea, a proposal, a note, etc.). Each object has a class and a property (attribute-value) list. There is a class definition language for specifying the structure, appearance, behavior, and interpretation of objects of each class and a markup language for laying out domain objects and other graphic objects on Tivoli pages. Thus, to fully tailor Tivoli to a particular style of meeting, one has to create domain class definitions plus import and export scripts.

For example, consider how domain objects might be used in the meeting in Section 2. There would be object classes for the scenarios and for group and category labels. Scenario objects would be imported from the spreadsheet. Participants would manipulate the scenario objects and create label objects in the meeting. The labeled and selected scenarios would then be exported back to the spreadsheet.

### Domain Object Definitions

We illustrate the way domain objects are defined and used by an example shown in Figure 3. Four domain object classes are defined in Figure 3a: Label, Task, RegionSum, and LinkedSum. The simplest is a Label object, which just has a name. A Task object has four properties: id, duration, name, and assignee. Domain objects appear on the Tivoli board as graphic objects, called *views*, whose appearances are defined in view layouts in their classes. A Task has two view layouts: view1 shows just its id and duration, whereas view2 also shows its name. Figure 3b shows 22 Task objects. They are all displayed as view1s, except for T8, which is displayed as a view2. The figure also shows four Label objects in four regions. Note that the fourth label, in the freeform region, has a scribbled name.

Figure 3a. Domain Object class definitions

```

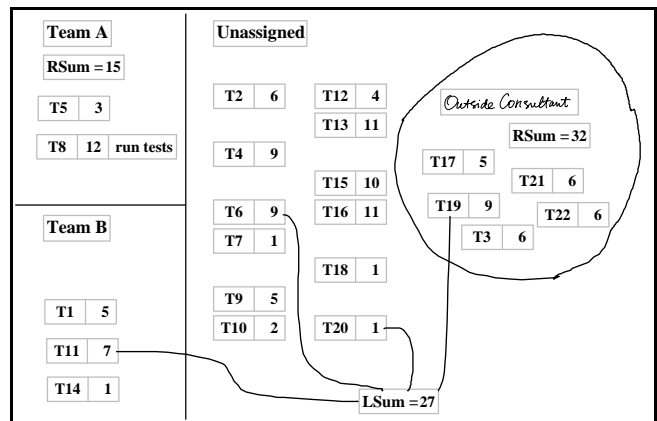
class Label (name)
  view1: name

class Task (id, duration, name, assignee)
  view1: id duration
  view2: id duration name
  computed value of assignee:
    get name of Label in my region
  actions:
    right in view1 → change to view2
    left in view2 → change to view1
    up on duration → duration = duration + 1
    down on duration → duration = duration - 1

class RegionSum (sum)
  view1: "RSum=" sum
  computed value of sum:
    add up durations of Tasks in my region

class LinkedSum (sum)
  view1: "LSum=" sum
  computed value of sum:
    add up durations of Tasks linked to me
  
```

Figure 3b. Domain Object instances on a Tivoli board.



Task objects have specialized behaviors in response to gestures on them<sup>2</sup>, as defined in the *actions* in Figure 3a. By making leftward or rightward gestures on Task views, the user can control how they are displayed. In Figure 3b, the user has “expanded” the view of Task T8 by a rightward gesture. Also, by gesturing upward or downward, the user can change a task’s duration value. Note that the first two gestures work on specific views, while the other two gestures work only on the duration field (i.e., gesturing up on the id field has no affect). The general form of an action rule in Tivoli class definitions is:

*view, field, gesture* → *action*

<sup>2</sup> Actually, on their views. It is often natural to refer to a view as if it were the object itself, and we do so frequently in this paper.

Objects for summing up durations are also defined in Figure 3a. A RegionSum object has a sum property, which is computed dynamically by a formula, which says to add up the duration values of all the Task objects in the local region. This computation is spatially localized to be relative to the spatial position of the RegionSum object. For example, there are two RegionSum objects in Figure 3b, one in the region labeled “Team A” and one in the region labeled “Outside Consultant.” The computed sums change dynamically as the user moves objects on the board between regions. For example, if Task T4 were dragged to the Team A region, the RegionSum there would jump to 24; and if the RegionSum object in the Team A region were dragged to the Team B region, it would show 13.

Tivoli supports computations based on spatial relationships — such as containment, location, relative position, and linkage — on a freeform whiteboard environment. For example, a LinkedSum object is shown in Figure 3b, which adds together the durations of Tasks linked to it. (More details about the mechanisms of spatial computation are presented in [14].)

Note that computations based on spatial relations *interpret the significance of the spatial arrangement of objects* on the board and encode them as values of object properties. A better example of this is the assignee property of Task objects, which is computed dynamically by finding a Label object in the same local region and getting its name property (see Figure 3a). For example, the assignee value of T5 is “Team A”; if T5 is dragged to the region below, its assignee value will change to “Team B”. If the name of any Label object is changed, then the assignee properties of Task objects in the same region are also changed. If these Tasks had been imported from a project management system, then these assignments could be exported back to the project management system.

The value of these mechanisms will be seen as we consider how this facility is used to create meeting tools.

## 6. MEETING SCENARIOS

We used meeting scenarios to drive the design and development of the domain object facilities. Based on years of experience in observing and supporting meetings with Tivoli, we chose several specific meeting scenarios. These meetings involved budgeting, decision making, presenting, reviewing, interviewing, and brainstorming. We created specialized meeting tools out of domain objects to support these scenarios. We also built tools to support generic meeting activities, such as managing agendas and timing, taking attendance, notetaking, recording actions, etc.

In this section we present a sampling of these to illustrate how domain object meeting tools can be integrated into the activities of a meeting.

### Budget Meeting Scenario

The tools for this kind of meeting are easiest to understand, because the behavior is like a “freeform spreadsheet.” This domain consists of cost items organized into budget centers, which are both represented by objects. The goal is to arrange the cost items to fit the budgets. See Figure 4.

Cost item objects have a succinct view, showing only a (truncated) name and cost. These are distributed among different regions, which are labeled with budget center objects. Doing a hold gesture on a budget center pops up an overlay view that documents the class. Figure 4 shows an overlay for the GIR budget center. It shows the four parts of a budget center object: a name, a computed sum of cost items in its region, a target budget, and a computed variance from the budget. It also shows that gestures can alter the target budget.

The meeting proceeds by manipulating cost item objects to eliminate the variance. Cost items can be moved between budget center regions and the lower row of regions, where they are not charged to any budget center. Note also that scribbled annotations are allowed, and they do not affect the computations.

There are additional computations that interpret the arrangement of objects. Each cost item has a budget-center property, which is computed by getting the name of the budget center object in the same column, and a charge-to property, which is computed by getting the name of the budget center in the same local region. Moving cost items to the lower row of regions result in a charge-to of nil, since there are no budget center objects in those regions.

### Decision-Making Meeting Scenario

Decision-making meetings are concerned with evaluating and choosing among alternatives of some type. In this scenario, the alternatives are projects; and the goal is to choose a subset of the projects to fund within the budget (the units are percentages of headcount).

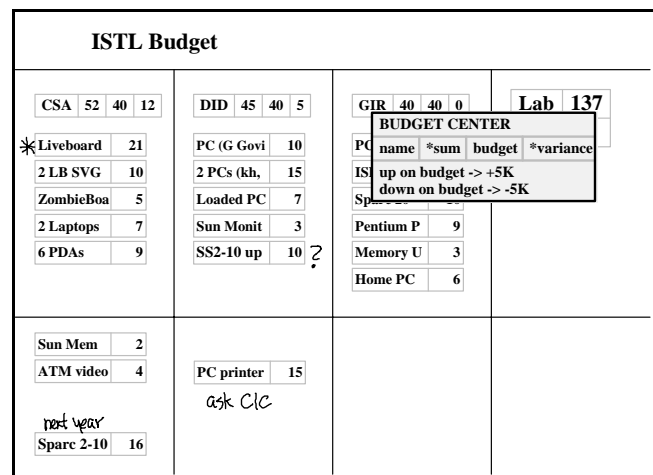
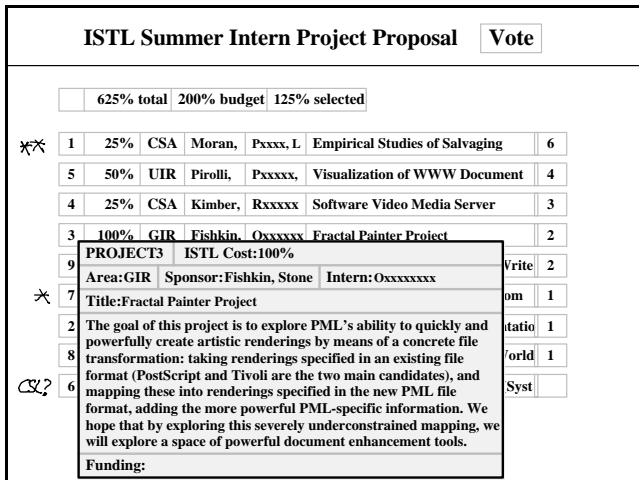


Figure 4. Tivoli Board for the Budget Meeting Scenario.



**Figure 5. Tivoli Board for the Decision-Making Meeting Scenario.**

A project object contains several properties. The list view of a Project object is a wide one-liner showing many of these properties. These can be arranged in a vertical list. See Figure 5. This meeting involves first reviewing the projects. A double-tap gesture on the list view of an object creates an overlay with an expanded view of the project object's data, as shown.

Freehand annotations can be made on the board, such as the stars marking certain projects in Figure 5. The project objects in the list can be manually rearranged: one simply taps on a project object to select it and drag it up or down. When released, the objects will adjust (opening and closing space) to keep it a neat list. The annotations move with the objects, because the implicit structure mechanisms in Tivoli work with *all* graphic objects on the board (see [14] for details).

It is sometimes useful to sort the projects. An up-down gesture on any field of a project causes the project list to be sorted according to that field. This sorting is animated so the users can better comprehend the action; and, again, the annotations are sorted with the project views.

In this scenario, meeting members vote on the projects. This is supported by vote objects. The "Vote" button at the top of Figure 5 is scripted to create a set of vote objects and place them to the right of each of the project objects, as shown. Votes are recorded by gesturing on the vote objects to increment (or decrement) the vote counts. Gesturing up-down on a vote object causes them to be sorted, carrying the project objects and annotations with them.

Finally, the choice of particular projects is recorded by gesturing on them. An up gesture on a project selects it, a single action that results in its status property being set to "selected", its view changed to red to visually distinguish it, and its cost added to the budget summary object at the top of the page. (Similarly, a down gesture deselects a project.)

## Generic Meeting Tools

The scenarios above illustrate that meeting tools vary from the specific to the generic (e.g., from project objects to cost item objects to vote objects). There are many generic elements to most meetings, and so we found it useful to define generic objects for these: agenda items, action items, people, notes, votes, labels, etc.

Usually the most specialized element of a meeting is the specific subject matter being discussed, such as the projects in the decision-making meeting or the scenarios in the meeting in Section 2. At the extreme level of generality we found it interesting to create a "thing" object that can represent any kind of thing being discussed. The generic thing object is useful, for example, in a brainstorm meeting where things get created by scribbling on the board. Scribbles can be encapsulated into a thing object by a box gesture. The things can then be arranged, marked with label objects, assigned to people objects, voted upon, etc.

Another generic element in meetings has to do with timing. We have a clock object that holds the current time. Our generic agenda items can be simple or timed. A timed agenda item has a planned time (which the user can set and adjust) and an actual time (which is computed by using the clock object). The agenda item object warns the users (with an audio action) when the actual time approaches the planned time.

Another use of the clock is in proportional timing. In a meeting that has to process a number of items (e.g., a conference program committee meeting), a timing summary object keeps track of how many items have been processed and computes and displays a prediction of when the meeting will end based on the current rate of processing.

Generic tools can be useful by themselves in some meetings, but they are probably more useful as starting points for tailoring. We next present a case study, where we started with generic tools and evolved a much more specialized and finely tuned set of meeting support tools.

## 7. CASE STUDY: IP REVIEW MEETINGS

For several years, we have been supporting the intellectual property process at PARC by providing Tivoli meeting support, plus audio capture, indexing, and salvaging tools [10, 13]. The central activity in this process (documented in [10]) is an ongoing series of meetings that review *invention proposals* (IPs) by rating and ranking them. After the domain object facilities were sufficiently complete and we had some experience designing both specialized and generic meetings with them, we designed a set of tools for the IP review meetings. These meetings have a rich structure and practice, which will become apparent as we describe the tools (also see [10]).

The workbase for intellectual property is Lotus Notes. We wrote a script that imports from Notes the working set of information needed for a given review meeting, defines a

set of domain objects, and lays them out onto several Tivoli pages. There is a page of attendees, an agenda page, pages for recording notes and ratings for each IP reviewed, a page for ranking IPs (called a *ladder*), and provision for dynamically creating new pages for notes on IPs not on the agenda. Being able to easily navigate within and between these pages is crucial, and Tivoli has a jump action that can be triggered by gestures (usually down and up for forward and backward jumps).

The main subject matter in these meetings are IPs, represented as IP objects, which consist of a dozen properties (about a page of textual information for each IP). The set of IPs need to be worked with in a number of different ways, and so many different views are defined: agenda views, title views (on review pages), small ladder views, plus larger views for displaying all of an IP's information on overlays (similar to Figure 5). Another reason for multiple views per object is that it makes it easy to propagate changing information, such as the ratings, to different pages.

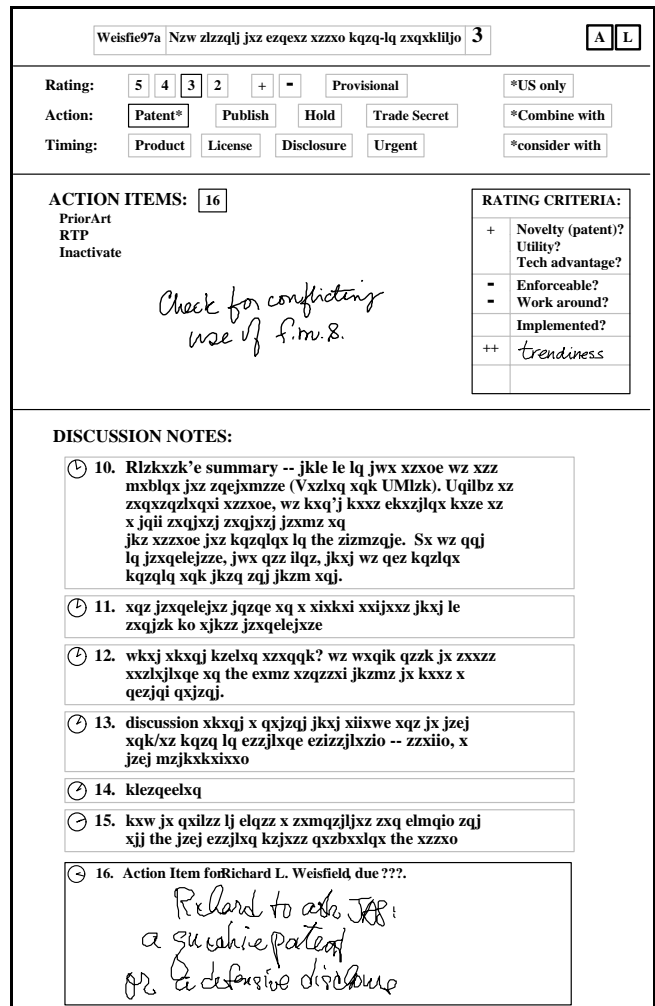
The agenda page is simply a list of IP agenda views (similar in appearance to the list of project views in Figure 5). The agenda can be edited, e.g., by making scribbled annotations or dragging IPs around to reorder the list; the navigation gestures follow the edited order. Gesturing down on an IP in the agenda jumps to a page for reviewing that IP.

An IP review page (Figure 6) contains four regions: a title region, a button region, an action item region for generating final actions, and a notes region for collecting notes and actions. The title region contains an IP title-view, which shows the rating value, and buttons to jump to the agenda and ladder pages. The rating is set by tapping on the various objects in the button region.

The main activity in the meetings is the discussion of each IP. The notes region supports the collecting of notes during the discussion. Notes can be scribbled in this region and converted to note objects. However, the preferred method is to take notes on a laptop and to beam them into Tivoli, which then creates note objects and appends them to the list of notes in the notes region. Notes can also be re-beamed if they have been further edited at the laptop. An interesting feature is that the beamed notes contain the times they were created (which is used later to control playback of the recorded audio [13]), and the note objects have a property for these times, shown as the little clocks in Figure 6.

The action item region is used at the end of the discussion to create any actions to follow up. Actions are recorded by scribbling in this region. Tapping on the Action Item label causes all scribbles in this region to be scooped up into a new action item object, which is appended to the notes region. The notes region is typically long and not visible on the board when actions are scribbled; thus a tiny view is also created in the Action Item region (see action "16" in Figure 6); gesturing on this view causes a full overlay to pop up in the Action Item region. An action item object has

**Figure 6. Tivoli Board with an IP Review Page.** Some text is "encrypted" to mask proprietary data.



a field for the "assignee." Tapping on this field brings up a menu of people to assign the action to. Alternatively, a person's name can be scribbled on the side and dragged onto the assignee field (which also causes a new person object to be created).

At the end of the meeting, it is often useful to gather all the action items generated during the meeting into one place to review them. The natural place for this is the Attendees page (Figure 7), which contains a list of person objects (the possible attendees). Gestures on these allow people to be marked as present. The names of unanticipated attendees can be scribbled and converted to person objects. A "Show Actions" button causes the current action items from the various review pages to be displayed as tiny views arranged next to the people they are assigned to. Gesturing on one of these causes an overlay to pop up (see Figure 7). An action can be reassigned by simply dragging it onto another person object (the action item object then automatically scoots to the end of the person object's row).

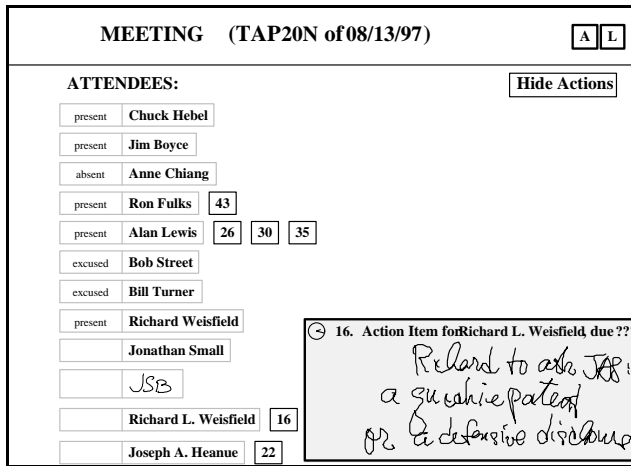


Figure 7. Tivoli Board with an IP Attendees Page.

Finally, there is the ladder page (Figure 8), which is a map of all current IPs in a particular technology area (not just the IPs being reviewed in the meeting). Small IP views are used so that an overview can be seen. Gesturing on an IP view brings up a full display of the IP on an overlay. The page is divided into regions with *ranking* label objects (“A”, “B”, “New”, “Hold”, etc.). IP views can be moved around by dragging, and Tivoli’s implicit structure mechanism maintains the list structure. An IP’s ranking is computed dynamically, as explained previously, from where it is positioned on this page (using both its region’s label and its position in the list). Scribbled annotations can be made anywhere on the page. Gesturing down on an IP causes a new page to be created, where notes on that IP can be taken, action items can be created, and/or its rating changed.

These tools were designed and evolved over a half dozen IP meetings starting in October 1996, and they have been in continual use since then, during which time only minor

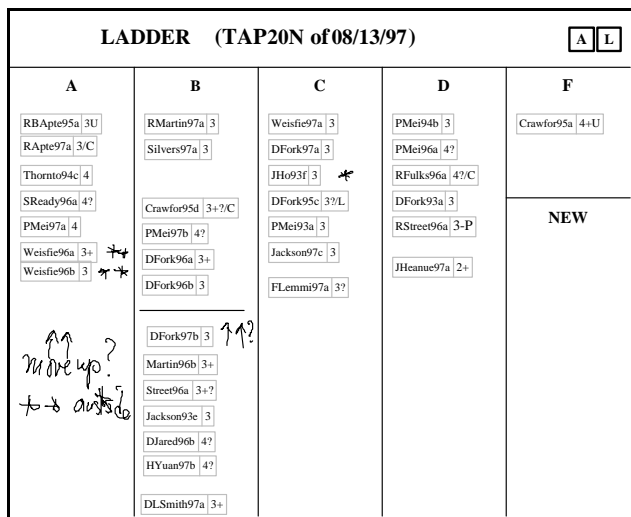


Figure 8. Tivoli Board with an IP Ladder Page.

changes have been made. The tools are finely tuned to the practices of these meetings. Further changes would involve changes in the practices. For example, we thought it would be useful to be more explicit about rating criteria, and so we experimented with a “Rating Criteria” object (shown in Figure 6); while useful in a couple of cases, it is not used most of the time.

These tools were developed in an “evolutionary engagement” [10] with the participants in the IP review meetings. Observations of the meetings and discussions with the participants show that the tools have resulted in several improvements in the meeting processes:

- The relationship among large sets of items is easily manipulated and explored, while all present can see and participate. Prioritizing IPs on the ladder is a qualitatively better activity. The rearrangable agenda enables a better-planned meeting.
- Information is retrievable when needed without being in the way when not needed. The details of older IPs can easily be popped up in the ladder page as needed.
- Information created or altered in one location is propagated to other locations. The ratings can be seen on several different pages; the action items can be gathered together for a final review.
- Notes can be taken and indexed on old IPs that are not on the agenda for the day. New review pages can be easily created from the ladder as the discussion spontaneously focuses on old IPs.

While all of these activities are supported with the tailored meeting tools, they exist in a whiteboard environment, where the meeting can always revert to scribbled notes when the specialized tools are not appropriate.

## 8. DISCUSSION

Now that we have described the domain object facility and how domain objects are used as meeting tools, we can step back and reflect on our objectives (listed in Section 3) in designing this facility.

### Connecting to the Workbase at the Semantic Level

Our first two objectives were to represent the semantic content of meetings and to connect to the workbase at this level of content.

It is apparent, as we show in Figure 2, that Tivoli can connect to the workbase by simply putting and getting files with editable page images. The problem is that this representation is not compatible with other file representations in the workbase, and thus any information has to be manually transcribed between representations. The connection is at the document (or page) level and not the object (or “item” or “field”) level of information. Tivoli has always been capable of having text elements on the board that can be freely moved around on the board. The problem is that the significance of this 2D arrangement is lost in converting it back into a textual representation (a string of characters).

Other systems, such as Lotus Notes, allow interaction with the workbase at the “field level” by having a database of “objects” that can be presented in different interactive views. But Notes does not offer a whiteboard-style view that can be used in meeting settings. The system that is closest to ours is Dolphin [18], which also uses the Live-Board. Dolphin offers a neat solution integrating the synchronous work in a meeting with the asynchronous work outside of meetings, but does this within a particular hypermedia model of nodes and links.

Our approach is to integrate with a variety of different workbases and to present a whiteboard-style user interface in the meeting setting. We do this by translating the data at an object level for presentation and by providing a facility to interpret the spatial arrangements created in the meeting. In this way, we can connect with any workbase that has an appropriate API at the level of objects.

### Designing Finely Tuned Tools

Our third and fourth objectives were to provide in the meeting setting a freeform whiteboard user interface that blends informal scribbling and annotations with structured domain objects. We have explored a variety of user interface techniques to integrate domain objects into Tivoli’s whiteboard environment, and we have developed some design principles for tailoring meeting tools with the domain object facility.

Domain objects behave fully like any other graphic objects on the board in Tivoli. In addition, they are responsive to special gestures. In tailoring such gestures we employ standard user interface design principles. Gestures are invisible and thus must be remembered. We found it helpful to use certain gestures consistently, e.g., the hold gesture for bringing up documentation of an object’s class behavior (as in Figure 4) (cf. [6]) and the double-tap gesture for bringing up further information about the contents of the particular object (as in Figure 5). To make gestures feel natural, the principle of “gesture compatibility” is relevant, such as an up gesture invoking an increment action, a right gesture invoking an expansion of the view in a rightward direction, or an up-down gesture invoking a sorting operation, where the objects in the list shuffle up and down as the sort is animated.

Scribbles and domain objects are well integrated. Users can, of course, scribble anywhere on the board around domain objects (as in Figures 4, 5, and 8). These objects and scribbles work together in Tivoli’s implicit structure mechanism. For example, the scribbled annotations in Figure 5 move with the project views when the project list is sorted. Scribbles made on the board can be incorporated into domain objects in various ways. Any set of scribbles can be converted to an object by a box gesture around the scribbles (the same technique as used in Dolphin [18]). Domain objects can be tailored to “absorb” scribbles that are dragged onto their views and to “pull” scribbles in their

spatial regions (which other systems cannot do). All these techniques allow users in meetings to easily shift between loose and structured modes of expression.

Our overall design strategy for using domain objects as meeting tools was to stress the use of board pages containing overviews of sets of data. This relies on the flexibility of the domain object view mechanism, allowing multiple views of differing detail. The principle for information presentation is to get as much information as possible on a page using smaller views (but allow users to pop up more information as needed), to use different pages for different arrangements of information for different purposes, and to allow easy navigation between these pages. The IP Review case study provides evidence that this is an effective strategy.

### Using and Interpreting 2D Spatial Arrangements

Our fifth objective was to allow the free use and interpretation of 2D spatial arrangements on the board.

Our strategy of providing overviews of information allows users to see the interrelationships among objects. By providing easy ways for them to rearrange and organize objects on the board they can explore new relationships. 2D spatial relationships — containment, adjacency, relative position, alignment, clustering, etc. — are the medium of expression and exploration. Our ability to have the system interpret some of these spatial relationships allows us to capture and encode semantic relationships that would otherwise be implicit. In doing this we are building on and integrating with the previously-developed spatial interpretation mechanisms in Tivoli [12, 11].

A distinguishing characteristic of our approach is the use of a range of 2D spatial relationships, as opposed to node-link structures. Dolphin [18] allows users to create relationships by expressing them in node-link structures and graph representations. This follows in the hypertext tradition, which stresses *explicit* links.<sup>3</sup> Indeed, in our own experience, from NoteCards [4] to Viki [8], it took us several years of experience and exploration to fully appreciate that 2D layout was usually a more expressive and more compact representation than node-link graphs and to understand that 2D spatial arrangements could be usefully interpreted [17], even though the relationships are more implicit and subtle.

### Tailoring Specialized Meeting Tools

Our final objective was to provide facilities for tailoring specialized meeting tools. We provide a scripting language for defining the structure, appearance, and behavior of domain object classes. Our language has most of the elements of the Oval [7] system (objects, view, agents, links), but in a much simpler form. Yet, in a similar way, we find

---

<sup>3</sup> So-called “idea processing” applications also focus on explicit node-link representations, e.g., the “mind maps” of Inspiration [5].

that this language allows for the creation of quite a broad range of meeting tools. Our facility is much more specialized than Oval in being integrated with a sophisticated whiteboard environment with spatial organization, manipulation, and interpretation capabilities.

Tailoring is a costly activity. It is possible to craft meeting tools from scratch. Our view is that one would begin with some generic meeting objects, such as those in Section 6 we used to create the IP Review tools (Section 7). We have identified a few generic objects, but there is a lot more to do to attain a polished set of meeting building blocks.

Most meeting support tools are geared to a particular task or process, such as gIBIS [1] for issue discussion, or a suite of different processes, such as GroupSystems [15] or MeetingRight [9]. These tools tend to be intentionally quite restrictive, so as to guide a normative process. Our philosophy is that tools are better adopted if they can be tuned to existing practices, at least initially. Building tools on a tailorable foundation provides the flexibility to do this. In our facility, tailoring must be done by reworking class definition scripts, which must be done between meetings; that is, tools are iteratively refined over a series of meetings. We have not provided any interfaces to allow class definitions to be edited in a meeting. Though this is a possible course of investigation, we do not think this is a good use of meeting time. Instead, we provide the flexibility in the user interface for working around the immediate limitations of the tools in a meeting by informal scribbling, thus providing an experiential basis for improving the tools. Our long-term experience in the case study in Section 7 has shown that this is an effective way to evolve both the tools and the meeting practices [10].

## 9. CONCLUSION

We have shown how to integrate structured domain objects into a freeform whiteboard user interface. This integration is useful, because it allows us to work in meetings in a whiteboard style with data from structured workbases. Further, these domain objects are tailorable so that they can be tuned to specific meeting practices. This increases the amount and quality of work that is possible in a meeting, as shown, for example, in our case study. These kinds of improvements are possible when tools can be tailored to well understood and predictable meeting practices. When meetings take unanticipated turns, users can slide into familiar informal methods of scribbling. The ability to do this depends on the smooth blending of structured and informal user interface techniques (which are discussed in more detail in a companion paper [14]).

**Acknowledgments.** We thank our colleagues in the Collaborative Systems Area of PARC for many discussions over the course of this work. Chuck Hebel worked closely with us in creating and evolving the tools for the IR review process. Thanks also to Beverly Harrison for critiquing this paper.

## REFERENCES

1. Conklin, E. J. & Begeman, M. (1988). GIBIS: A hypertext tool for exploratory policy discussion. *Proceedings of CSCW'88*, 140-152.
2. Denning, A. *ActiveX Controls Inside Out*, 2nd ed., Microsoft Press. 1997.
3. Elrod, S., Bruce, R., et al. (1992). LiveBoard: A large interactive display supporting group meetings, presentations, and remote collaboration, *Proceedings of CHI'92*, 599-607.
4. Halasz, F. G., Moran, T. P., & Trigg, R. (1987). NoteCards in a nutshell. *Proceedings of CHI'87*.
5. *Inspiration: the thought processor*. (1990). Ceres Software, Inc.
6. Kurtenbach G., Moran, T. P., & Buxton, W. (1994). Contextual animation of gestural commands. *Proceedings of Graphics Interface'94*, 83-90. Canadian Information Processing Society.
7. Malone, T., Lai, K., & Fry, C. (1992). Experiments with Oval: a radically tailorable tool for cooperative work. *Proceedings of CSCW'92*, 289-297.
8. Marshall, C. C., Shipman, F. M., & Coombs, J. H. (1994). VIKI: Spatial hypertext supporting emergent structure. *Proceedings of ECHT'94*.
9. *MeetingRight software user's guide*. (1994). Rochester, NY: Xerox Quality Solutions.
10. Moran, T. P., Chiu, P., Harrison, S., Kurtenbach, G., Minneman, S., & van Melle, W. (1996). Evolutionary engagement in an ongoing collaborative work process: a case study. *Proceedings of CSCW'96*, 150-159.
11. Moran, T. P., Chiu, P., & van Melle, W. (1997). Pen-based interaction techniques for organizing material on an electronic whiteboard. *Proceedings of UIST'97*, 45-54.
12. Moran, T. P., Chiu, P., van Melle, W., & Kurtenbach, G. (1995). Implicit structures for pen-based systems within a freeform interaction paradigm. *Proceedings of CHI'95*, 487-494.
13. Moran, T. P., Palen, L., Harrison, S., Chiu, P., Kimber, D., Minneman, S., van Melle, W., & Zellweger, P. (1997). "I'll get that off the audio": A case study of salvaging multimedia meeting records. *Proceedings of CHI'97*, 202-209.
14. Moran, T. P., van Melle, W., & Chiu, P. (1998). Spatial interpretation of domain objects integrated into a freeform electronic whiteboard. *Proceedings of UIST'98*.
15. Nunamaker, J., Dennis, A., Valacich, J., Vogel, D., & George, J. (1991). Electronic meeting systems to support group work. *Communications of the ACM*, 34(7), 40-61.
16. Pedersen, E., McCall, K., Moran, T. P., & Halasz, F. (1993). Tivoli: An electronic whiteboard for informal workgroup meetings. *Proceedings of INTERCHI'93*, 391-389.
17. Shipman, F. M., Marshall, C. C., & Moran, T. P. (1995). Finding and using implicit structure in human-organized spatial information layouts. *Proceedings of CHI'95*.
18. Streitz, N., Geissler, J., Haake, J., & Hol, J. (1994). Dolphin: integrated meeting support across local and remote desktop environments and liveboards. *Proceedings of CSCW'94*, 345-358.
19. Trigg, R., Moran, T. P., and Halasz, F. G. (1987). Adaptability and tailorability in NoteCards. *Proceedings of Interact'87*.