VIDEO ARTIFACTS FOR DESIGN: BRIDGING THE GAP BETWEEN ABSTRACTION AND DETAIL

Wendy E. Mackay, Anne V. Ratzer & Paul Janecek

University of Aarhus
Department of Computer Science
34 Åbogade
8200 Århus N, DENMARK
{mackay,avratzer,pjanecek}@daimi.au.dk

ABSTRACT

Video artifacts help bridge the gap between abstraction and detail in the design process. This paper describes how our use and re-use of video artifacts affected the re-design of a graphical editor for building, simulating, and analyzing Coloured Petri Nets. The two primary goals of the project were to create design abstractions that integrate recent advances in graphical interaction techniques and to explicitly support specific patterns of use of Petri nets in real-world settings.

Using a participatory design process, we organized a series of video-based design activities that helped us manage the tension between finding useful design abstractions and specifying the details of the user interface. Video artifacts resulting from one activity became the basis for the next, facilitating communication among members of the multi-disciplinary design team. The video artifacts provided an efficient way of capturing and incorporating subtle aspects of "Petri Nets In Use" into our design and ensured that the implementation of our design principles was grounded in real-world work practices.

Keywords: Coloured Petri Nets, Design abstraction, Design process, Marking menus, Participatory Design, Scenario-based design, Toolglasses, Video artifacts, Video Brainstorming, Video Prototyping

INTRODUCTION

Video is a powerful tool that can be used throughout the design process, from initial observation of users, through idea generation (video brainstorming) and design exploration (video prototyping) to system evaluation. Mackay and Tatar (1989) present an early collection of different uses of video as a research and design tool. More recent examples have highlighted innovative uses of video to support critical incident analysis (Hartson and Castillo, 1998) and to examine quantities of video data (Lange et al., 1998, Buur & Søndergaard, 2000). While various multimedia data analysis systems have been developed over the years (Halasz et al., 1987, Mackay & Davenport, 1989, Hibino & Rundensteiner, 1998, Mackay & Beaudouin-Lafon, 1998), the problem remains that video is generally considered too detailed and cumbersome to work with. Even those who strongly advocate gathering detailed observations of use seek ways to abstract their findings, e.g., Ericsson & Simon's (1993) Verbal Protocol Analysis and Beyer and Holzblatt's (1998) Contextual Inquiry. How can we take advantage of this highly contextual, qualitative data to generate usable design abstractions without losing the key details?

DIS '00, Brooklyn, New York.

We have been exploring how to use video artifacts throughout the design process to help us manage the tension between qualitative details and design abstractions. Our video artifacts act as both the output of one design activity and the input to the next. This reuse provides an efficient way to identify relevant material and apply it in later design phases. Going back and forth between detail and abstraction ensures that our design principles are appropriately grounded and that the design details are organized in a conceptually useful and accessible way.

This paper describes how we use video artifacts to support the design of a new tool for creating and simulating Coloured Petri Nets. We begin with a brief description of the CPN2000 design project and then explain the four key elements of the design framework: the specific interaction techniques in the user interface, the design principles that guide the technical design, the generalized contexts of use and the users' individual patterns of interaction. We then explain our design process, which addresses both the tension between abstraction and detail and the tension between technology and use. Finally, we trace the evolution of a particular design concept, a "styleglass", showing how video artifacts support communication and help us manage issues relating to both abstraction and detail. We conclude with a discussion of the role of video and video artifacts in the design process.

The CPN2000 Project

Coloured Petri Nets or CPNs (figure 1) are a graphical formalism used by researchers and practitioners to describe, simulate and prove complex concurrent systems (Jensen, 1992). The Design/CPN tool, developed in the 1980s at the University of Aarhus, provides a graphical editor to interactively create, simulate, and analyze CPNs. The tool has been very successful and is in active use by over 600 organizations around the world. However, the time has come to update it, taking advantage of advances in user

interface design and CPN simulation over the past decade.

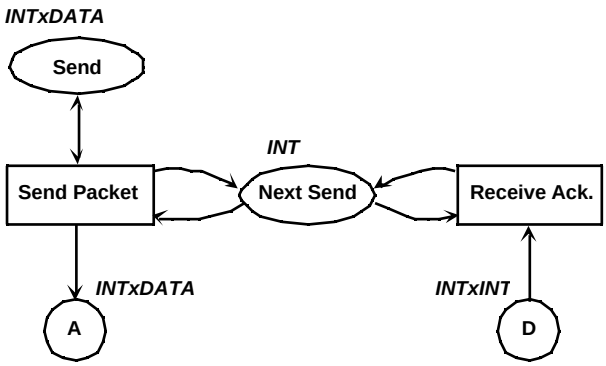

Figure 1: A simple Coloured Petri Net

The CPN2000 project is a complete redesign of the Design/CPN tool, including radical changes to the user interface. The project draws from three areas of expertise within the University of Aarhus computer science department: Coloured Petri Nets, human-computer interaction and object-oriented languages. The project, funded by CIT, Hewlett-Packard and Microsoft Research, began in February 1999 and involves participants from each research area, with a core group of 11 people responsible for design and implementation.

A primary goal of the redesign is to explore how recent advances in graphical user interfaces and interaction techniques could improve the CPN editor's support for Coloured Petri Net designers. This requires an understanding of "Petri Nets In Use", our term for the collection of guidelines, work styles, and interaction patterns that influence the way designers build CPNs (Janecek et al., 1999). We used a cooperative design process (Greenbaum & Kyng, 1991) with users (in our case, Coloured Petri Net designers) contributing actively in the evaluation and redesign of their own "Petri Nets In Use".

DESIGN FRAMEWORK

The design of any interactive system involves work within two domains: The *technology domain* consists of the factors that influence the architecture of the system, its functionality and the interaction techniques. The *use domain* consists of design guidelines, overall work styles and the individual patterns of use. Addressing the tension between these domains has been a cornerstone of Human-Computer Interaction research for the past two decades (Card, Moran and Newell, 1983, Norman & Draper, 1986, Norman, 1988).

Our design framework highlights two main issues in each domain. Within the technology domain, we work out the details of specific *interactiontechniques* as well as create abstract *design principles* that unify the architecture and conceptual model of the new tool. In the use domain, we examine the detailed *interaction patterns* or ways in which CPN designers work as well as create abstract user models that capture the *context of use*, in this case the concept of "Petri Nets in Use". Note that both domains involve an inherent tension between abstractions, which ensure coherent design, and details, which ensure that the system actually works in the real world. Good design requires both: The challenge is to integrate the two, so that they enhance rather than compete with each other.

Figure 2 organizes the four main considerations of our design framework (interaction techniques, design principles, context of use, and interaction patterns) along these two axes: technology/use and abstraction/detail. The next section describes each in greater detail.

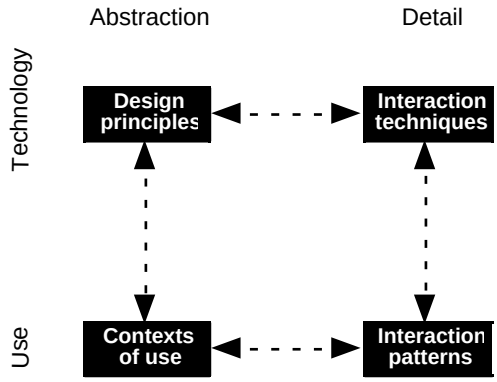

Figure 2: The design framework

Interaction Techniques

The first consideration in the design framework is technological: the choice of interaction technique (upper right-hand corner, figure 2). We are working with the Instrumental Interaction model (Beaudouin-Lafon, 2000) in which instruments mediate the interaction between a user and the objects in the interface. This model extends Shneiderman's (1983) description of direct manipulation interfaces, which follow three principles:

- **continuous representation** of the objects of interest with meaningful visual metaphors,
- **physical actions** (such as presses of labeled buttons) instead of complex syntax, and
- **rapid incremental, reversible operations** whose effect on the object of interest is immediately visible.

Most traditional graphical interfaces implement direct manipulation through a combination of Windows, Icons, Menus, and Pointing (WIMP). Although these have a number of strengths, e.g., when designed well they are self-revealing to a novice user, they often distract users from the object of their work and force them to shift their focus to intermediate command objects, such as menus and dialog boxes. In the context of a graphical editor, this separation between object and action is inefficient and slow.

Toolglasses (Bier et al., 1993), are a good example of a "post-WIMP" interaction technique (Kurtenbach et al., 1997). They are floating, semi-transparent tools that support instrumental interaction, taking advantage of direct, two-handed manipulation. For example, a color palette toolglass, allows the user to apply a color to or

absorb a color directly from an underlying object. She uses the dominant hand to move the section of the toolglass containing the desired color over the object of interest and then uses the dominant hand to apply the color by "clicking through" the toolglass onto the underlying object. The user can thus specify both the object and the action with a single mouse click.

Standard architectures for graphical interfaces do not support most post-WIMP interaction techniques nor are they trivial to add to existing interface toolkits. One of our major design activities has been to create an architecture that lets us explore a variety of these techniques. At first we believed that we could choose a single "best" interaction technique, such as toolglasses or marking menus (Kurtenbach and Buxton, 1994). However, our empirical studies showed us that users in different use situations or work contexts have different preferences among these techniques.

We conducted a controlled experiment that asked users to try each of three different interaction techniques (toolglasses, marking menus and floating palettes) on tasks that were identical except for the work context. (See Mackay et al., 2000, for a preliminary report.) We found that, while mechanically correcting a net from hand-written notes may involve exactly the same actions as thinking through the problem and making corresponding changes, the user's perception of the task, and their preference for interaction techniques, are quite different. No single interaction technique is always superior (or inferior) to the others. Some individuals strongly prefer one interaction technique in most circumstances (preferences are approximately evenly distributed across the three techniques). Other users switch their preferences according to the particular context of use.

These findings convinced us that we needed to design the tool to integrate several interaction techniques into a single user interface and make it easy to swich among them. In order to do this effectively, we needed to articulate a set of unifying design principles that would guide our design choices. This led us to the second consideration of the design framework (upper-left corner, figure 2).

Design principles

Three design principles emerged during the design process. They began from the initial insights about instrumental interaction and were influenced by our interest in integrating multiple interaction techniques. The videotapes of actual use also influenced the design principles by clarifying the different contexts in which the system would be used, and highlighting the diversity of approaches in the current tool. The principles, defined in greater detail in Beaudouin-Lafon & Mackay (2000), are:

1. **Reification**, which creates interaction objects that represent concepts in the interface. Thus, commands can be made accessible as instruments, combinations of properties can be turned into styles, and the selection of multiple objects can be tagged and accessed as groups. For example, the alignment command can be reified into a guideline object, an instrument that actively maintains the layout of graphical objects on the screen.

2. **Polymorphism**, which extends the power of commands with respect to interface objects, allowing similar operations to be applied to a variety of objects. For example, objects can be cut, copied or pasted, any operation can be undone, and operations that apply to a single object can be applied to groups of objects.

3. **Reuse**, which provides a way of capturing and reusing patterns of use. The user can reuse both previous commands (input) and responses from the system (output). For example, "redo" and "undo" commands reuse input whereas command macros reuse output. New commands may be created from existing commands, using a partial list of pre-defined arguments.

The reification principle has strongly influenced the design of the new tool. For example, in the earlier tool, the user aligns a set of objects by applying a "vertically align center" command. To add additional objects, the user must reselect the aligned objects. In the new tool, the alignment command is reified into a magnetic guideline, a visible, first-class object that is continuously accessible and modifiable. New objects can be attached to the guideline, and moving the guideline also moves all the objects attached to it.

These design principles are even more powerful when combined. For example, they led us to the insight that we do not need the concept of selection, which in turn makes it possible to combine and integrate four interaction techniques: direct bi-manual interaction, marking menus, floating palettes and toolglasses. In order to understand how best to combine them from a user's perspective, we explored abstractions about use, the third consideration in the design framework (lower-left, figure 2).

Context of Use

Early in the project, we realized that we needed to distinguish between the creation of pre-defined Petri Nets and the more pragmatic problem of creating Petri Nets in everyday work settings. We explored an abstraction we call "Petri Nets in Use", i.e. the collection of factors that influence the way a CPN designer works. We include visual representations and work styles that vary according to context.

Visual representations are essential to a CPN designer's work: they communicate the underlying meaning of the net. Guidelines, such as Jensen's (1992) readability guidelines, include suggestions on the use of structure and graphical attributes to emphasize the flow of data and the semantic relationships among objects. CPN designers draw from other sources as well, including information visualization techniques (Noik, 1994) and guidelines from graph drawing research (Di Battista, 1999). Understanding these guidelines and how designers employ them is essential if we are to create a usable design tool.

Work styles describe the CPN designer's overall process of creating a net and depend upon the context of use. In general, our interviews and field observations showed that CPN designers usually begin by modeling the core functionality of a net and then progressively add details, switching between editing (building the model) and simulation (testing the model). This is similar to Schon's (1983) description of design as a process of "seeing", "moving" and "reflection-in-action". We identified three qualitatively-different CPN design activities, reflecting three distinct *contexts of use*:

1. **Edit an existing Petri Net:**
   Implementing a set of hand-written changes on-line requires little active understanding of CPNs, but does require skill with purely graphical or layout changes to the net.
2. **Modify an existing Petri Net:**
   Copying a working Petri Net and modifying it to address a new problem requires thinking about Petri Nets as nets, not simply their graphical components, but lets the user take advantage of existing layouts and already-debugged code.
3. **Create a new Petri Net from scratch:**
   Creating a new net from scratch requires the deepest understanding of CPNs, but need not require any attention to layout, which may be done later.

Most current tools are designed to support making final layout decisions (the first context of use). The assumption is that the CPN designer already knows what the net should look like. Our field studies showed that this use context usually occurs after group design meetings. One person is assigned to update the on-line net based on notes taken during the meeting. The second use context is more common. CPN designers rarely start with a blank page, instead they prefer to reuse previously-solved problems. The main difficulty with this strategy is that it favors experienced CPN designers who have built up a personal library of usable nets; novice CPN designers with the least knowledge and ability are the most likely to be forced to create new nets from scratch, the least common, but still important, context of use.

Abstracting these three contexts of use enables us to identify, separate, and understand the reasons behind many of the interaction patterns we see in our videotapes. This leads to the fourth consideration in the design framework (lower-right, figure 2), the detailed analysis of individual interaction patterns.

Interaction patterns

Interaction patterns describe the low-level sequences of commands performed by CPN users as they build, edit, or simulate CPNs. Users perform particular combinations of commands based on the context of use as well as the interaction techniques available to them. We can compare a set of interaction patterns and examine how they change under each context of use. For example, a CPN designer, responsible for implementing a set of graphical changes written on paper, might turn seven objects blue, rearrange five of them along a line, then add three new objects and add arcs connecting them. In this context, he would prefer an interaction technique that facilitated a "command-first, object-second" syntax, such as a tool palette.

Another CPN designer might perform the identical set of commands when modifying an existing net, but perform them in a different order, modifying each object in turn. She would prefer an interaction technique like a marking menu, to facilitate an "object-first, command-second" syntax. A third CPN designer who was creating a new net from scratch could also perform the identical set of commands, but in a different order. He would think about the structure of the net and systematically work his way around the cycle. He would benefit from an interaction technique like a toolglass, that made it easy to switch between object-command and command-object, facilitating rapid creation of a new CPN.

Of course, the user interface of any tool also shapes the users' interaction patterns. For example, Design/CPN users quickly learn to anticipate future actions so they can perform the same command numerous times and reduce the high overhead of switching tools. Identifying and characterizing these individual interaction patterns requires access to many examples of individual use of the tool, under a variety of contexts. Understanding how these interaction patterns obtain in the existing tool helps us determine which should be supported in the new tool, which must be improved, and which could be omitted entirely.

DESIGN PROCESS

Figure 3 shows how our design process maps onto the design framework in figure 2. We begin by observing users (1), then brainstorm new ideas (2), narrow down those ideas into a workable design (3), and finally evaluate the design (4). Of course, this cycle is iterative: we conduct additional field studies, brainstorming, prototyping and evaluation sessions as needed.

The small gray boxes in figure 3 represent the video artifacts collected in that step. The arrows illustrate the dual role of each collection of video artifacts. They both influence our understanding of the four main components of the design framework (Interaction Techniques, Design Principles, Contexts of Use and Interaction Patterns) and they also directly affect the collection and interpretation of later video artifacts: Video clips and related artifacts, such as storyboards, are not only the output of individual design activities but also serve as input to subsequent design activities.

Video clips taken from the field studies provide a framework for the brainstorming sessions. Video clips of brainstormed ideas, together with clips of real-world examples of use, inform the design sessions. The resulting video prototypes pose questions that guide our evaluations, including formal experiments, informal user studies and long-term studies of use in the field. Finally, video results from the different forms of evaluation suggest directions for subsequent field studies, identify new issues that require additional brainstorming sessions and provide answers or justifications for particular design decisions.

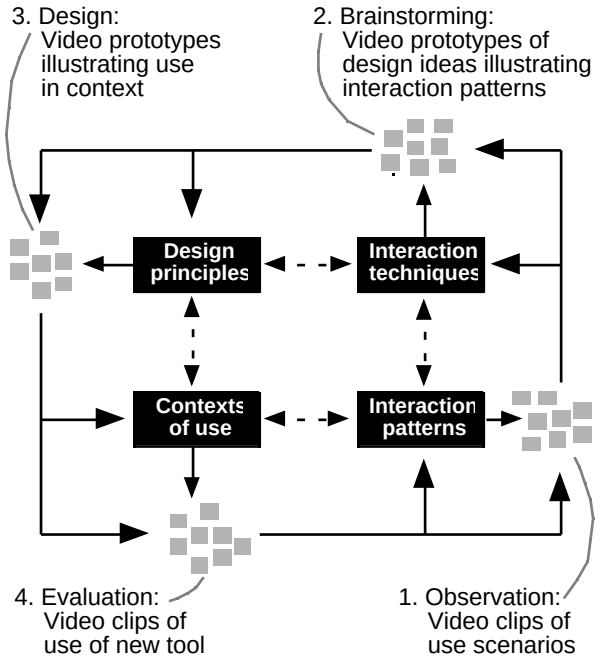

Figure 3: Video artifacts in the design process

Recycling video artifacts is efficient: By continually re-evaluating the video in different contexts, we achieve both a deeper understanding of the design problem and become increasingly familiar with the details of both the technology and its use.

VIDEO ARTIFACTS IN THE DESIGN PROCESS

After some initial skepticism, video has become an essential and accepted component of our design process. Video from each activity strongly enhances communication among members of the design team (who have diverse backgrounds and knowledge of CPNs) and with the project sponsors. The next section describes each phase of the design process, with specific examples of the types of video we collect and how it can be reused in later design phases.

Phase 1: Observation

We began the project by finding out as much as we could about use of the existing Design/CPN tool. We observed and videotaped expert users from both the local CPN group and from a local company, as well as student users learning the tool. We also interviewed members of each group. We transcribed the videos, translating from Danish into English where necessary and highlighting interesting events and problems with the tool. We also reviewed the tapes several times to identify "normal" or recurrent patterns of use.

The first type of analysis simply gathered video clips together to illustrate the most common user interface problems. In each case, we selected clips of 30-60 seconds that illustrated a particular problem. We edited them together, ensuring that each problem began with an expert user, who sometimes, but not always, performs the task well, followed by student or novice users struggling with the interface. For example, all users had difficulty managing the "red boxes" that are created by the simulator to display its current state. The boxes were very small and usually placed on top of the object that was the subject of the red box, obscuring it. When users tried to enlarge and move the red boxes, they would often miss and inadvertently move another part of the net, requiring the net to be re-edited. Other common problems included: creating an object called a guard, following hypertext links to retrieve error messages, creating an arc, editing text, using "undo" to recover from a mistake, and negotiating with a particular dialog box that specifies a simulation step called “binding”.

Another type of video analysis, based on activity theory, examined the focus shifts between the task at hand and the interface (Bødker, 1996). For example, in one four-minute clip of students trying to import a text report, we see 19 shifts of attention between the Petri net, the text report file, the text box, the windows, the menus, the dialog boxes and the prompts. Some users spend so much time shifting between the tool and the activity that they lose track of what they were attempting to do in the first place. We created a summary video of both types of analysis and presented it to the design group. Although some situations were very familiar to the CPN developers, others surprised them: expert users seriously underestimate how much time they spend on minor manipulations of the tool, especially those involving layout.

The third type of video analysis involved the creation of *use scenarios* (Mackay & Bødker, 1994, Carroll, 1995) to create compressed, but real, illustrations of patterns of use. We created storyboards that illustrated a series of activities that would be performed by a real user, in a real context. For example, we selected a collection of video clips of expert users from our field study of a small firm that uses the Design/CPN tool. We arranged the clips into scenarios that illustrate activities the new tool must accommodate: rearranging the structure of an existing net, checking syntax and port assignments, modifying the styles in a telephone protocol, changing a colour set, and inserting a new component of a net. We created storyboards with 8-16 elements. Each element represented a video clip with a still image ("best frame") to the left and a text description of the context and the activity to the right. We then edited video clips together into two- to four- minute scenarios. We gave members of the design team the corresponding storyboards, so they could follow the story and take notes while watching the video scenarios.

Figure 4 shows several segments of a paper storyboard used to construct one of these edited video clips. The clip shows the modification of a simple protocol, including creation of a new counter and modification of the corresponding graphical attributes. The series of images, taken from the video, illustrate the changes made throughout the task, while the accompanying text descriptions provide details about the precise operations performed and any difficulties encountered.

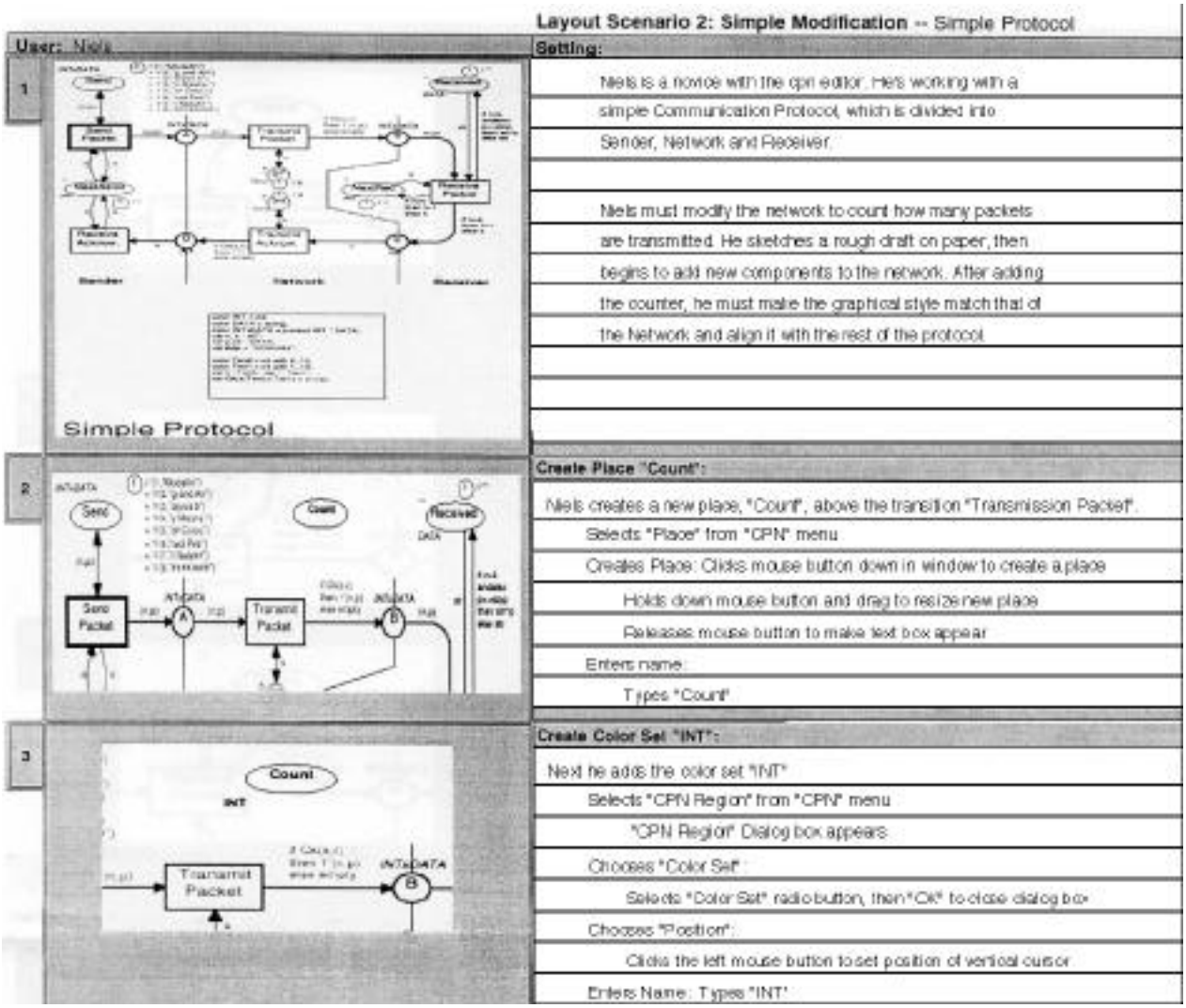

Figure 4: A section of a storyboard with a user scenario

For some scenarios, we used actual video from our field studies. When the existing video was not sufficiently clear, we re-taped the situation exactly as in the field studies. These video scenarios proved useful for a variety of design activities. They helped us abstract and generalize important issues that the new tool must address while remaining grounded in the details and actual context of use. They served as a method of interpreting and analyzing the data in terms of work practices while serving as inspiration for later design activities. The video clips and interviews also helped specify required functionality for the new tool and deepened our own understanding of "Petri Nets in Use".

The observation phase produced two forms of reusable video. The short clips, illustrating currently-open design issues, e.g., ineffective interaction patterns, inspired and stimulated the brainstorming sessions in phase 2. The longer, more complex use scenarios, which showed an entire task being performed in a particular use context, focused and grounded the design process in phase 3.

Phase 2: Video brainstorming

The purpose of phase 2 is to generate as many new ideas as possible, without evaluating them. We performed diverse brainstorming activities with different groups of people. We began with a large workshop that included members of the local CPN user group. We asked them to identify good and bad characteristics of the old tool and generate ideas for the new tool. Later, we held weekly or bi-weekly design sessions for more focused brainstorming activities. (These sessions always included several CPN users.) Sometimes the topics were open, but more often we addressed issues raised from the field studies or from the design sessions. We noticed that the format of the brainstorming session deeply affected both the quantity and quality of the resulting ideas. We engaged in four types of brainstorming:

1. **"Say it"**
   A traditional brainstorming exercise, in which participants describe in words (verbally or on cards) as many ideas as they can, each of which is written on the whiteboard for later analysis. This approach generates the largest quantity of ideas, but they are often vague or poorly formulated.

2. **"Show it"**
   Participants draw on the whiteboard to illustrate their ideas. This results in a reasonable quantity of ideas that are somewhat better formulated, but they are often too abstract and "static"; they rarely

provide much insight into the details of the interaction.

3. **"Act it"**
   Participants use simple prototyping materials including colored pens, paper, transparencies, Post-it notes, and paper versions of real Petri Nets, to help "act out" each interaction idea. User interface elements are drawn or cut out as needed and participants demonstrate the interaction to work out what the user would actually do. This results in fewer ideas, but each is better thought out and is more likely to capture the dynamic nature of the interaction. Acting it out also facilitates communication within the design team.

4. **"Videotape it"**
   Participants use the same prototyping materials as above but act out each idea in front of the video camera (figure 5). Once participants are familiar with the technique, they generate ideas almost as quickly as in the "Act it" style. Although this brainstorming technique produces the fewest number of ideas, they are usually the most detailed and programmers can use the resulting video clips to create software prototypes. Participation is also high: it is difficult to sit quietly when everyone else is preparing for a new "take".

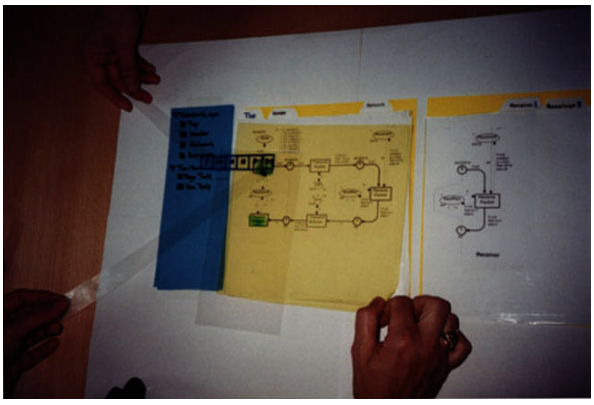

Figure 5: A video brainstorming session

The first sessions in which we tried the "act it" and "videotape it" styles of brainstorming required more time and some participants wondered if it was worth the trouble. However, these soon proved to be our most productive and enjoyable design sessions.

We continue to use all four forms of brainstorming. The traditional "Say It" style works well with large groups and requires no preparation. The "Show it" style works well in the context of an ordinary meeting, when a design question comes up and it makes sense to quickly brainstorm alternatives. However, for most pre-scheduled brainstorming meetings, we bring a box of prototyping materials and one or more video cameras, to make it easy to "Act it" and "Videotape it". Although we reuse ideas from each type of brainstorming session, the videotaped ideas are the most likely to be remembered and have the greatest influence on later design activities.

Phase 3: Design

Phase 3 involves the difficult process of narrowing down the variety of design options generated in phase 2 and selecting those that will appear in the new tool. Our approach involves both top-down and bottom-up activities. The former involves the creation of a functional table that systematically identifies all the necessary and desired functions of the new tool. Within each category, e.g., layout, navigation, and creation, we identify the objects and their corresponding operations. Then, for each function, we identify a set of possible interaction techniques. We also link design scenarios, which include detailed interaction patterns, to the functional table. This helps us analyze how well the new design supports Petri Nets In Use and provides an overview of the components required for the new tool.

The bottom-up activities include video prototyping sessions designed to illustrate how the new tool might be used. These sessions differ from brainstorming in that the goal is to negotiate a single design, not generate different options. For each particular design problem, we begin by reviewing video clips of relevant use scenarios and brainstorming ideas and then develop a design scenario. We alternate between whiteboard-based discussions and prototyping sessions with paper-based mock-ups, periodically referring back to the design principles. Then subgroups each create video prototypes of their design scenario, working through the details of how a user would interact with that particular design.

Team members enjoy playing with "special effects" such as starting and stopping the camera to create the illusion of system feedback. A more subtle technique involves a second "live" video camera as a "Wizard of Oz" tool (Chapanis, 1982). The wizard has access to a set of prototyping materials representing screen objects, such as CPNs and toolglasses. Other team members stand by, ready to help move objects as needed. The live camera is pointed at the wizard's work area and the image is projected onto the TV monitor in front of the user. The user can interact with the different objects that appear on the screen; the wizard moves the relevant paper materials in direct response to each user action. The other camera records the interaction between the user and the simulated software system on the screen.

This is a particularly powerful video prototyping technique because it gives the "user" a real sense of what it might actually feel like to interact with the proposed tool, long before it has been implemented. Seeing a video clip of someone else interacting with a simulated tool is more effective than simply hearing about it; but interacting with it directly is more powerful still. The programmers develop software prototypes based on the video clips from the video prototyping design sessions. Both the video clips and the resulting software prototypes can be evaluated in a variety of ways.

Phase 4: Evaluation

Phase 4 provides feedback about the success of our design ideas. Of course, a systematic experimental evaluation of all possible design options is impossible.

The design of the new tool is too complex and each decision affects numerous others. So, evaluation often involves returning to previous design phases to gather more information and video data. For example, at a CPN workshop, we asked six pairs of CPN designers who had not been involved in the tool design to first try and then comment on each of three interaction techniques. We showed videotapes of our design scenarios and then presented software prototypes of the new tool, derived directly from the video brainstorming videotapes.

As mentioned earlier, we also conducted a formal experiment to compare the different interaction techniques under different use contexts. We asked 18 experienced CPN designers from outside the project to work through seven scenarios, derived from our observations of expert users in an external company. The scenarios were varied systematically to capture each of the contexts of use. The video artifacts gathered from the observation sessions inspired the overall design of the experiment and provided the basis for some of the scenarios used in the different experimental conditions.

Most recently, we conducted a 6-week field study. We provided a CPN user with version 1 of the new tool and asked her to record her reactions in a daily log. We also recorded all her keystrokes in a form that we can replay. At the beginning and again at the end of the study, we videotaped her as she worked through a current problem and showed what she liked and did not like about the new tool.

Our videotapes of CPN users during these evaluation exercises gave us additional insights about patterns of use. For example, we are currently working on the problem of managing groups of objects. The evaluation data allowed us to identify several strategies for creating and managing groups, in different contexts.

The entire project is tracked via a web server that allows users to edit and create new pages (see Beaudouin-Lafon, 2000). We include minutes from all design meetings, as well as results from our data analysis. We also maintain a video archive, which makes it easy to go back and ask questions as they arise. For example, we refer back to the video clips of prototypes made at different stages, which helps us track changes in the tool, remember ideas that would otherwise get lost and check for consistency across versions.

DESIGN PROBLEMS AND SOLUTIONS

We claimed that video artifacts gathered in one phase provide important input to later phases of the design process. This section gives an example, tracing the evolution of the styleglass through the reuse of video artifacts.

One of the first issues raised in our interviews and brainstorming sessions was the problem of managing attributes, such as color and line thickness. Effective use of these graphical attributes enhances readability of the net, helps make semantically-similar objects look the same, emphasizes differences among functionally-distinct objects and highlights the main flow of a CPN cycle. CPN designers want to be able to create personalized style sheets and search for objects based on their style attributes.

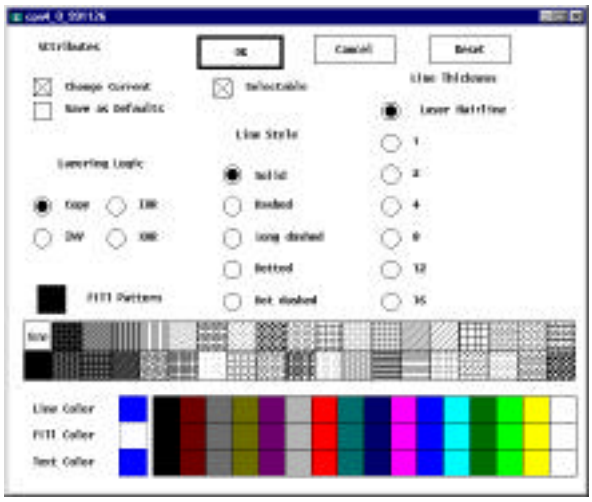

Figure 6: Managing attributes in the old tool

Figure 6 shows how the former tool, Design/CPN, manages graphical attributes with a standard, modal dialog box. Like most commercial software applications, the user is forced to choose among a large number of attributes. The "ideal" process has four steps: select the object, choose the "set graphical attributes" entry in the menu, select the desired attribute in the dialog box, and then click "OK". Repeating this process for multiple objects is cumbersome and users usually find it more efficient to copy an object with the desired style and then modify the non-shared attributes.

We examined numerous video clips of users trying to change attributes with the current tool. We also looked for examples in which users expressed their personal style preferences and when they were required to modify attributes to meet corporate guidelines. For example, one company uses line thickness rather than color to specify different types of network flow, because they only have a black-and-white printer.

Video brainstorming Workshop

We selected a range of video clips concerning attribute changes and presented them to the design team. We asked them to brainstorm ideas for a "styleglass", a toolglass that would let users specify styles in a lightweight, natural way. We also identified a set of video clips of innovative new interaction techniques that could facilitate use of styles, i.e. toolglasses, zoomable and mark-based interfaces.

We faced a dilemma when planning the initial brainstorming session. On the one hand, we were eager to expose the members of the design team to alternative user interface techniques. On the other hand, we did not want to simply pick our favorites and impose them upon the group. We decided to show individuals different videos of these interaction techniques, at least a week before the first brainstorming session. The goal was to seed new ideas, but not make them the driving force in the brainstorming sessions. This worked well: the developers drew ideas from their own experiences,

e.g., an emacs-style interface for editing CPNs, as well as from the new techniques. The group considered a wide variety ideas and rejected most of them, including, to our regret, gestural input and zoomable interfaces.

The most radical idea to be explored by the design team was the toolglass. They were intrigued by the use of two hands because they thought it might significantly reduce the number of keystrokes and frustration involved in changing attributes. However, they were unsure about the coordination required between the two hands. Prior to the first two-hour video brainstorming session, design team members watched a compilation of video clips of users struggling to change attributes with the old tool and everyone saw the CHI '94 videotape of Toolglasses and Magic Lenses (Bier et al., 1994).

We invited a large number of CPN experts to join us and divided the group into three multi-disciplinary teams. Each group watched a different video clip of a use scenario and generated ideas on how to use a toolglass to improve the interaction. Figure 7 shows a design idea drawn on a transparency from a subgroup exploring a CPN layout scenario, the first "styleglass".

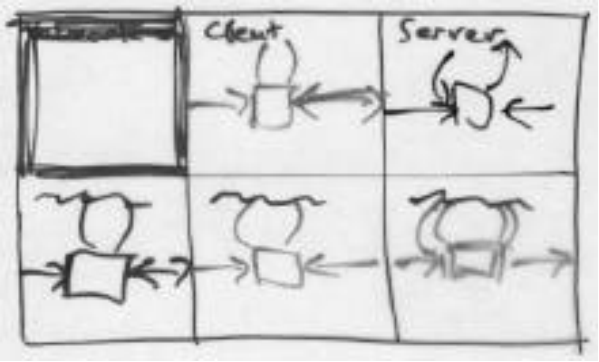

Figure 7: "Styleglass": Style preview toolglass

The styleglass allows the user to display objects in the net with different styles: one style for the client side of the network, one for the server side. Other group members explored variations, including one-handed styleglasses, styleglasses that can be resized, and styleglasses that permit cutting and pasting of graphical structures.

The three design groups brainstormed from three to 11 video illustrations each. (The group with three video clips had a number of ideas in each clip, whereas the groups with more ideas created a separate clip for each individual idea.) The resulting video clips proved to be an extremely efficient method of communicating to members of the programming team who had not participated in the video brainstorming sessions. One programmer implemented several of the most interesting ideas over a couple of days, so that the entire design team was able to experience their "look-and-feel" at the next week's meeting. We later systematically modified the characteristics of these software prototypes, changing the number and color of the cells, switching between one and two-handed input, and switching between transparent and opaque cells, in order to better understand the details of the interaction.

Once members of the design team had a chance to actually try the styleglass, they were convinced of its potential and decided to pursue it as a fundamental part of the design. This was not a minor decision: users of the new system would be required to use both a mouse and a separate pointing device, such as a trackball. However, with the growing popularity and lower cost of USB devices, it seemed worth the risk.

Once the design team members saw a direct link between the video artifacts they had created and working software, they were more open to subsequent video brainstorming and prototyping sessions. The video archive helped designers judge how faithfully the software reflected their original design ideas and helped keep track of ideas that would otherwise have been lost. This built confidence in both the video artifacts and the software prototypes and significantly enhanced communication among CPN designers and software developers.

Design

The design phase involves making choices among design alternatives. After referring back to the videos of the original use scenarios, the video brainstorms and the software prototypes, the design team began the process of video prototyping. Each subgroup discussed ideas for a design scenario that would capture a real-world example of using the new tool. For example, one group explored ideas for a "search and replace" styleglass that would make it easy to find and change items with a certain properties or combination of properties.

Based on these ideas, each group was given an empty version of the storyboard in figure 4 and asked to create a design scenario, with sketches and descriptions of how a user in a real-world situation would use the tool. Unlike the video brainstorming exercises, which allowed individuals to pursue their own ideas, video prototyping exercises require group consensus. Team members must discuss and negotiate the ideas before coming to agreement on a common solution. Discussions of the abstract principles of reification, polymorphism and design both helped to generate new possibilities and to choose among design alternatives.

Design scenarios require relatively long sequences of steps which are separated with short, explanatory title cards, as in a silent film. This facilitates editing in the camera and also makes the subsequent video clip easier to understand. In most video prototyping sessions, each design team would create a five to ten minute video clip, videotaped directly from the storyboard. As with video brainstorming, the fact that the group had to collaborate and act out the interactions in front of the camera. increased the overall level of participation and communication among team members.

Figure 8 shows one of the ideas for a search-and-replace styleglass used in a video prototype. The circular styleglass has three sections around the outside to specify line style, line thickness, and line color. It is drawn on a transparency and placed over a pre-printed net, to show what it would mean to "click through" and

change a graphical attribute. In this design, the center area contains the current settings of each of the different sections of the styleglass, and highlights other objects in the diagram with the same attributes. A related styleglass allows the user to specify the values of attributes to search for and the attributes to replace them with. This would make it possible, for example, to set the styleglass so that it finds all objects that are red and makes their outlines dashed.

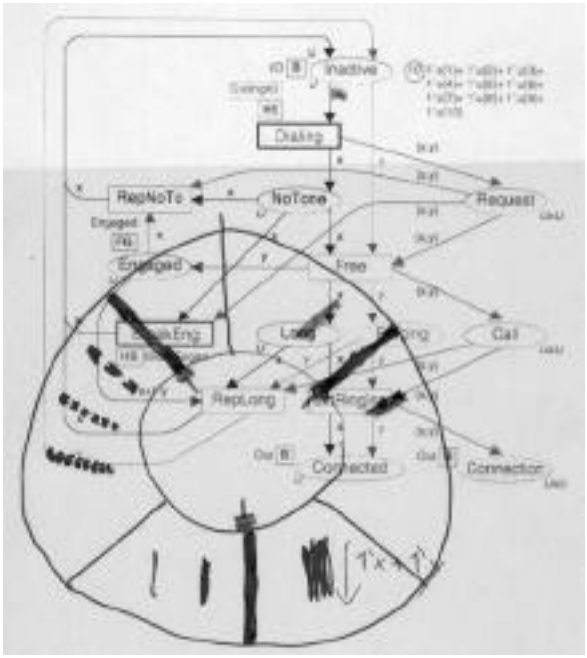

Figure 8: Graphical search and replace styleglass

Evaluation

We continued to refine and evaluate the low-level interactions with styleglasses by testing them with the use scenarios. We also used the design principles to ensure consistency among design ideas and to generate new avenues for exploration. We conducted design walkthroughs with both design team members and outside CPN users, using video clips as source materials. We ran several studies of the interaction details, i.e. reaction time, styleglass appearance, the use of two hands and the contexts of use.

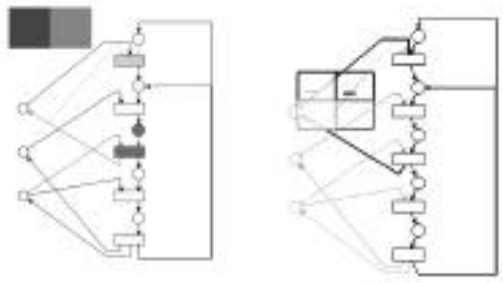

Figure 9: Styleglass prototypes.
(a) Click through Styleglass.
(b) Magic Lens version of Styleglass.

For example, we presented the prototypes described in the previous section to a dozen CPN users at a CPN workshop. We videotaped their use of the different design solutions and asked them which they preferred. We added the video clips of these observations to our video archive and used them to help develop the second major set of software prototypes.

Figure 9a shows the version that a user clicks through to apply an attribute to an underlying object. Figure 9b shows a Magic Lens version with transparent cells showing the net underneath in different styles and a click-through technique for applying the styles. The upper two cells of the styleglass show the objects in different line thicknesses and the lower two cells show the arcs in different colors.

Figure 10 shows the most recent version of the styleglass, in version 1 of the new CPN tool. The user can change the color, dash pattern or line thickness of an object by "clicking through" the appropriate cell in the top four rows. The last row has cells for picking up styles from existing objects and applying them to other objects. The user can carry as many as four personalized styles, to use as needed. This design reduces the number of steps required for applying an attribute, from four in the old tool to one in the new tool.

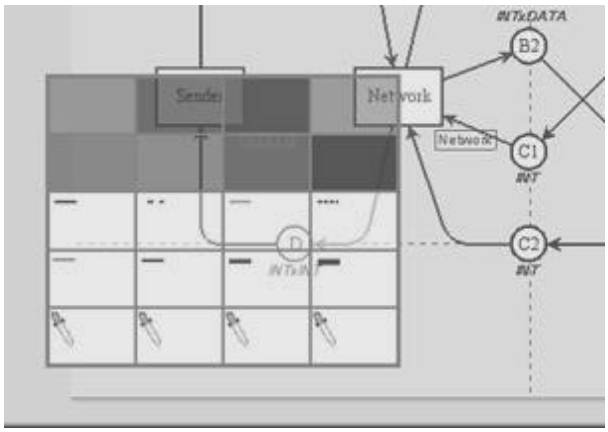

Figure 10: Attribute styleglass in the new tool

The specific appearance of and interaction with the styleglass is based on our on-going examinations of the video artifacts produced during the different design activities and continuously asking users for feedback. Although it is easy to see the connection between the first idea sketched in figure 6 and the final version shown in figure 10, many of the details have changed.

The role of the video camera

The video camera plays different roles throughout the design process. During the design process, the camera can serve as an unobtrusive "fly on the wall" to collect information about users at work. In contrast, the presence of the camera directly affects participant's behavior during video brainstorming and prototyping sessions; focusing their attention, increasing their awareness of the details of the interaction, and facilitating communication among the participants.

We were surprised at how differently participants acted between the "act it" (without the camera) and the "videotape it" video brainstorming exercises. Knowing

that ideas will be captured and viewed later encourages participants to think through each idea and really work out the details. Also, preparing an idea so that it can stand alone on a videotape requires teamwork: several people must manipulate the materials together to give the illusion of the user's interaction from the camera's perspective. This increases the required level of communication and improves the understandability of each idea. Similarly, the camera's presence actively changes the behavior of participants during video prototyping. The combination of the written storyboard and the video tape helps the team members create a coherent sequence of activities that is grounded in a real-world context, but illustrated in a way that can be implemented directly with the software.

Most of the local CPN users were not active participants in the weekly brainstorming or design sessions and only saw the results of their video brainstorming ideas after an interval of six months. They were pleased to see the direct link between their early brainstormed ideas and the working version of the new tool. Several commented that it was not until then that they finally understood the purpose of the video brainstorming sessions.

CONCLUSION

Our design process exploits the use of video artifacts to capture and communicate the details of how users interact with software. We use these artifacts to manage the tension between abstract discussions of design principles and detailed discussions of the user interface. One can deride observations of use in the field as merely "anecdotal evidence", which may or may not be typical of ordinary use. Yet, we have found that even small quantities of video examples shown to system designers may surprise them and challenge their assumptions about how users use existing systems. Video clips of external users, ranging from novice to expert, help counterbalance the often strongly-held opinions of the CPN experts participating in the tool's design.

While we cannot hope to capture every aspect of use, we can capture the most important aspects of both usual and unusual situations, providing a rich and concrete basis from which to design. A potential danger in using many unrelated video clips is the temptation to create separate solutions to deal with each individual problem. Abstract design principles that apply in many situations help us to uncover the more general problems hidden in specific examples and provide simpler, more elegant solutions that address a wider range of needs.

Video artifacts may serve multiple roles in any design process. They capture not only the basic functions of the software, but also more subtle considerations of the software as it is used in real-world contexts. By recycling video artifacts, we can move between activities that stress the specifics of the interaction and those that explore the general principles underlying the design, integrating the two and bridging the gap between abstraction and detail.

ACKNOWLEDGMENTS

Our thanks to all the members of the CPN2000 design team who participated in our design process, including members of the Beta, HCI, and CPN groups. We would also like to thank the CPN users, students, and experts who participated throughout the design process. Thanks especially to Michel Beaudouin-Lafon for comments on an earlier draft of this paper.

REFERENCES

Beaudouin-Lafon, M. (2000) Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. In *Proceedings of Human Factors in Computing Systems, ACM/CHI2000*, the Hague, the Netherlands, ACM Press, pp. 446-453.

Beaudouin-Lafon, M. (2000) The TWIKI interactive web server. *DAIMI Technical Report*, University of Aarhus. (http://www.daimi.au.dk/~mbl/twiki)

Beaudouin-Lafon, M. and Mackay, W. (2000) Reification, Polymorphism and Reuse: Three principles for designing visual interfaces. In *Proceedings of Advanced Visual Interfaces, AVI 2000*, Palermo, Italy, pp. 102-109.

Beyer, H. & Holtzblatt, K. (1998) *Contextual Design. Defining Customer-Centered Systems.* Morgan Kaufmann Publishers.

Bier, E., Stone, M., Pier, K., Buxton, W., and DeRose, T. (1993) Toolglass and magic lenses: The see-through interface. In *Proceedings of the 20th annual conference on Computer graphics,* pp. 73-80.

Bier, E., Stone, M., Fishkin, K., Buxton, W., and Baudel, T. (1994) A taxonomy of see-through tools. CHI '94 video, In *CHI'94 Conference Companion,* p.225.

Bødker, S. (1996). Applying Activity Theory to Video Analysis: How to Make Sense of Video Data. In Human-Computer Interaction. In Nardi, B.A. (ed.) *Context and Consciousness - Activity Theory and Human-Computer Interface.* The MIT Press. (pp. 147-174)

Bødker, S. (1996). Applying Activity Theory to Video Analysis: How to Make Sense of Video Data. In Human-Computer Interaction. In Nardi, B.A. (ed.) *Context and Consciousness - Activity Theory and Human-Computer Interface.* The MIT Press. pp. 147-174.

Buur, J. & Søndergaard, A. (2000) *Video Card Game: An augmented environment for User Centered Design discussions,* In *Proceedings of DARE 2000, Designing Augmented Reality Environments.* Elsinore, Denmark. ACM/SIGCHI, pp.63-70.

Chapanis, A. (1982) Man/Computer Research at Johns Hopkins, *Information Technology and Psychology: Prospects for the Future.* Kasschau, Lachman & Laughery (Eds.) Praeger Publishers, Third Houston Symposium, NY, NY.

Card, S., Moran, T., and Newell, A. (1983) *The Psychology of Human-Computer Interaction*. Hillsdale, N.J.: Lawrence Ehrlbaum Associates.

Carroll, J. (1995) *Scenario-based design. Envisioning work and technology in system development.* NY: Wiley & Sons.

Di Battista, G., Eades, P., Tamassia, R., & Tollis, I. (1999) *Graph Drawing -- Algorithms for the Visualization of Graphs*. Prentice Hall, New Jersey.

Ericsson, K.A. & Simon, H.A. (1993) *Protocol Analysis: Verbal Reports as Data.* Cambridge: MIT Press.

Greenbaum, J. & Kyng, M. (1991) *Design at work*: *Cooperative design of computer systems*. NY: Lawrence Erlbaum Associates, Inc.

Halasz, F., Moran, T. & Trigg, R. (1987) Notecards in a Nutshell. In Proceedings of CHI+GI '97, Human Factors in Computing Systems. Toronto, Canada, ACM Press. pp. 45-52.

Hartson, R. and Castillo, J. (1998). Remote evaluation for post-deployment usability improvement. *Proceedings of ACM AVI '98, Conference on Advanced Visual Interfaces.* Bari, Italy: ACM.

Hibino, S. & Rundensteiner, E. (1998) Comparing MMVIS to a time-line for temporal trend analysis of video data. In *Proceedings of ACM AVI '98, Conference on Advanced Visual Interfaces*. Bari, Italy: ACM. pp.195-204.

Janecek, P., Ratzer, A., & Mackay, W. (13-15 October 1999) Petri Nets in Use: Redesigning Design CPN. In *Proceedings of the Second Workshop on Practical Use of Coloured Petri Nets and Design/CPN*. (K. Jensen, Ed.). Aarhus, Denmark, pp.119-131

Jensen, K. (1992) Coloured Petri Nets -- Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts. *EATCS Monographs on Theoretical Computer Science*, Springer-Verlag.

Kurtenbach, G. & Buxton, W. (1994). User Learning and Performance with Marking Menus. In *Proceedings of ACM Human Factors in Computing Systems, CHI'94*, ACM Press, pp.258-264.

Kurtenbach, G., Fitzmaurice, G., Baudel, T., & Buxton. W. (1997). The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency. In *Proceedings of ACM Human Factors in Computing Systems, CHI'97*, ACM Press, pp.35-42.

Lange, B.M., Jones, M.A. & Meyers, J.L. (1998) Insightlab: An immersive team environment linking paper, displays, and data. In *Proceedings of ACM CHI '98 Human Factors in Computing Systems*. Los Angeles, California: ACM/SIGCHI, pp. 416-423. pp. 550-557.

Mackay, W.E. and Davenport, G. (July 1989). Virtual Video Editing in Interactive Multi-Media Applications. *Communications of the ACM*, Vol. 32(7), pp. 802-810.

Mackay, W.E. and Tatar, D. (1989) Workshop on Video as a Research and Design Tool. Special Issue of the *SIGCHI Bulletin*. ACM/SIGCHI.

Mackay, W.E. and Bødker, S. (1994) Workshop on Scenario-Based Design. In *CHI'94 Conference Companion.*, Boston, MA: ACM Press.

Mackay, W. & Beaudouin-Lafon, M. (1998) DIVA: Exploratory Data Analysis with Multimedia Streams. In *Proceedings of ACM CHI '98 Human Factors in Computing Systems.* Los Angeles, California: ACM/SIGCHI, pp. 416-423.

Mackay, W.E, Beaudouin-Lafon, M., Ratzer, A.V. and Janecek, P. (2000) The effect of work context on the use of three interaction techniques. *DAIMI Technical Report*, University of Aarhus. http://www.daimi.au.dk/~mackay/publications.html

Norman, D.A. and Draper, S.W. (1986). *User-Centered System Design*. Hillsdale, New Jersey: Ehrlbaum Associates.

Norman, D. A. (1988) *The Design of Everyday Things*. New York: Basic Books.

Noik, E. G. (1994) A Space of Presentation Emphasis Techniques for Visualizing Graphs. In Proceedings of *Graphics Interface* '94, pp. 225-233.

Schon, D. (1983) *The Reflective Practitioner*. New York: Basic Books.

Shneiderman, B. (1983). Direct Manipulation: a Step Beyond Programming Languages. *IEEE Computer*, 16(8), pp. 57-69.