Ethnographic Data Collection and Analysis for Product Development: Our Experiences with a Collaborative Process

Scott Lewis, Michael Mateas, Susan Palmiter, & Gene Lynch

Abstract

We have developed a process for using ethnographic data to drive design in a product development environment. This process involves three main steps: collecting observational data, analyzing the data to produce a model useful for design in a technical domain of interest, and successfully communicating the results of this analysis to engineers, marketers, management; in effect, all project team members. For each of these three steps, we detail our approach and experiences with the process, discuss the artifacts and models that we produced, and present the tools used.

Introduction

What is the appropriate role for user interface specialists in advanced development organizations? As members of the advanced development organization in Tektronix, Inc., this was a central question for us, and a question that is not satisfactorily answered by examining the current role of user interface specialists in other contexts. Neither the academic study of user interface design, nor the traditional role for user interface specialists in product development provides a good model for the role of the user interface specialist in an advanced development organization, where the goal is to provide a useful input directly to product engineering teams engaged in actual product definition, design, and development. Such input is rightfully expected to be timely and prescriptive, so that product development teams know what technology to build to deliver customer value.

Traditionally, the discipline of user interface design in product development is more proscriptive than prescriptive. Consider, for example, the typical product development process for a new hardware or software product. Usually, a researcher, engineer or marketeer will generate a product or design idea, get support for the idea among peers in the company and then get approval for pursuing the product idea, including marshalling the appropriate resources for design, development, testing and deployment. Commonly, any usability testing that occurs (or even serious thought about the how the product will be used in context) will take place well after many fundamental system design decisions have been made. At this point in the product development process only relatively small scale changes in the system can be made; two common (almost cliche) examples are menu nomenclature or menu item ordering. As valuable as such improvements are for the ultimate success of the user interface, the design of the system behavior (the basis of the actual dialog with a user) is mostly fixed by the definition of the system architecture, typically completed well before any usability testing begins. Such designs are usually uninformed by precise knowledge of the use domain, user's tasks or the usage context, and therefore provide more of a reflection of the concerns of a set of technologists than those of the ultimate user of the system.[1]

The traditional, proscriptive role for user interface specialists puts them in an unnecessarily weak position for influencing product design. Even with rapid prototyping at the earliest phases of a project, followed by continuous iterative design and testing, the input to the system designers from this process is not prescriptive. That is, such input does not, fundamentally, give much guidance to system designers about what they should build. At best, it constrains the design space for system designers, and simplifies the search for an optimal (or acceptable) design. At worst, it provides negative feedback about the current design and does little to constrain design alternatives, putting the designer back to the beginning of the search for an acceptable design. What is lacking is a set of design constraints that are based upon a full understanding of the domain, and the actual context of system use. We believe that a lack of constraints on system design from precise knowledge of a user, her domain, and context of use is a major flaw in the typical product development process and results in system design that is driven more frequently by technical or political concerns than user or context of use concerns.

An alternative role for user interface specialists is one of taking responsibility for constraining and making system design decisions very early in the product life cycle. This would involve data collection at a high enough level to drive very high-level design choices (e.g. "what product should be built for this market?", "what functional concept

[1] Note that the input from the UI research community typically is also not sufficiently prescriptive for product design, because it is frequently focussed on more general (non-domain specific) questions of human-computer interaction; making it difficult or impossible to apply to many design decisions in a specific design domain.

should this system embody?", "what should the UI metaphor be for this system?", "what system architecture fits our knowledge of the use domain?"), suggest multiple design vectors and effectively constrain the design space with knowledge of the user's domain. As design alternatives are explored and found more or less desireable given the knowledge of the domain, these design choices should be explicitly related back to the domain knowledge, so that these constraints can be passed on to other design team members during and after the life-cycle of the project.

In this role, what is now called user-interface design is more accurately described as user-and-domain-informed system design. We believe that such a role should include the following specific responsibilities:

1. Collect data to constrain design in the domain of interest. As we describe below, we feel that modified ethnographic techniques provide a means to collect these data in an engineering context and provide concrete constraints upon the system design well in advance of product concept.
2. Analyze and synthesize the data to aid other project team members in understanding the domain of interest. In short, given the data collected in 1) we feel it is important to produce a model of the user's domain that can be used by product development organizations to effectively drive system design throughout the entire product life cycle.
3. Produce actual system designs that are driven by a knowledge of the user, her tasks and the domain of use. This responsibility places the full weight of design squarely on the shoulders of the user interface specialist, rather than on some other system designer (i.e. software designers). We feel this is where the responsibility for system design belongs, as the user interface specialist in advanced development is in a unique position to gather and apply information about the user, the user's domain, and knowledge of appropriate technical constraints to generate, develop and evaluate system designs.
4. Communicate user data, models and system design ideas to engineers. We feel that an important role for user interface specialists in advanced development is to communicate their work to product development teams.

Blomberg, Giacomi, Mosher and Swenton-Wall [ ] have suggested similar strategies for incorporating ethnographic data into system design. What follows is a description of our team's implementation of this approach for a recent advanced development project at Tektronix Laboratories and our positive and negative experiences with the process. First we describe the domain of interest for this project to provide the appropriate application context for our work. Then we describe our effort to collect data about a domain of interest using ethnographic techniques (modified for use in an engineering context, as described below), analyze the data collected, produce a model of the data useful for constraining system design, generate system designs from our analysis, communicate those designs to product team members outside the advanced development organization at Tektronix, and evaluate the effectiveness of our approach.

Design Domain

The domain of interest for this work is video editing. Editing video (for television shows, commercials, movies, special productions, etc) is currently a very time intensive, sometimes tedious, multi-person enterprise. The work typically includes hands-on input from a large cast; including video editors, directors, and producers, and often includes input from graphic artists, sound artists, special effects artists, and animators. These roles frequently collaborate heavily and continuously during the editing to make and implement real-time decisions about the content, shape, character, sound, and look of a video production. These collaborations currently almost always occur with the roles co-located in both space and time. The expense of bringing and keeping these many professionals together during the editing process is considerable. As well, by necessity this process is wasteful of participant's time, as there are many occasions where one role (e.g. director) must wait for another role to complete a task (e.g. completing an edit) before work can continue. One thought from our research group was that much of the need for physical co-location could be eliminated by allowing collaborators to work remotely while providing the video (and other production artifacts) over a network to support the collaboration appropriately.

The idea of collaborative networked video presented several novel user interface issues and many questions. The video industry is rapidly evolving from one of a linear, analog, tape-based environment to one that is totally digital and disk-based. This transformation requires a thorough understanding of the old paradigms of editing and how they will transfer to newer media. For example, with tape-based video editing the time code on a video tape was the key to finding a piece of needed video. With disk-based access, immediate random access to video clips is available.

Yet with this access, the time codes have become meaningless and naming the video scenes became critical, but tedious.

Work that previously could only be performed in sequence, since there was only one master video tape source, can now be performed in parallel due to the random access nature of digital media. This is conducive to video editing since given that many people contribute to the construction of a time-based media, it is now possible (e.g.) to allow assistant editors or directors to cue up good takes of a given scene while the video editor works on a totally different portion of the final program--adding some potential for parallelism to a highly serial process.

Another novel feature of the domain is the characteristics of the individuals involved in the work. The editors, directors and producers typically have very well defined roles, but are very artistic and creative individuals. The interface designed for them must reflect their heavy reliance on the skills of their collaborators as well as their individual artistic expression.

Terminology

Producer: Sponsor for the video work. Oversees all aspects of the production.

Director: Artistic coordinator for the production. Often will have assistant directors to assist on various aspects of production.

Video Editor: Works hands-on with the video editing equipment. Helps to shape video in conjunction with objectives of the director and adds artistic qualities.

Scene: Segment of script that may have one or more video clips associated with it.

Take: One possible video clip to be used in a scene. There may be several takes of the same scene with varying acting style, camera angle, or motions.

Video Clip: Segment of video that will be used in a scene. Typically uses only one camera angle.

Data Collection

Finding that our current methods were inappropriate for understanding this domain, we turned to observational analyses methods that have their basis in ethnography. Ethnography is steeped in a tradition of detailed evaluation of observational data [12]. Table 1 presents a comparison of the traditional usability testing methods and direct observational analysis techniques.

In observational analysis typically there are no a priori assumptions about the underlying structure of the work. Instead, it is an exploratory analysis to detect structure--a form well suited for domain and task analysis. This approach also lends itself to gathering data useful for new product design and integration of that product into the current work environment.

Although well suited to domain analysis, observational analyses come at a high cost. The analysis time to sequence time (AT:ST), or time to analyze the video tape as compared to the real length of the video, can range from 5:1 to 100:1 for observational [10]. These ratios are due to the granularity of analysis. Discourse analysis, where the conversation in the video tape is scrutinized at the breath and pause level [4], would require a precision demanding extremely detailed examination. This level of precision makes these analyses time prohibitive for most industrial settings. In addition, the expertise necessary to carry out these analyses makes them unsuitible as a potential engineering tool.

Such a fine-grained analysis may also make it difficult to detect activities taking place at a higher level. For example, while analyzing eye movements and gesturing, it may be difficult to abstract the user's task process or overall work flow. In addition, the link between observational analyses and design directives in research is often far from clear. This makes these techniques less desirable since for us data analyses must contribute directly to system design.

Exploratory Sequential Data Analysis and Interaction Analysis. From this analysis of the strengths and limitations of our Directed Dialog method and the observational techniques, we developed our own process borrowing concepts from Exploratory Sequential Data Analysis (ESDA) [10] and Interaction Analysis [5]. The generic ESDA process for collecting and analyzing the data is outlined in Figure 1.

We chose to use observational analysis techniques because the domain was new to us and network-based collaborative video editing was at the time a new system concept with no current products in the market. Using this more intensive and in-depth technique allowed us to both understand the current processes in place during video post-production and judge how new technology would impact those interactions.

After observing our first session of two video editors and a director working together on a promotional video tape, we quickly realized that unstructured observation was inadequate and inefficient for the numerous video editing sessions, each 2-4 hours in duration, that we planned. Although we videotaped the session, we had no consistent or reliable pointers into that source tape. To be time effective, we devised a coding scheme that served as pointers into the raw video tape.

Three main roles, with specific coding duties for each, were defined for each of three observers. One observer recorded overall observations and impressions, the second interactions, and the third events. Interactions are a specialized form of events that merited our attention because of the collaborative nature of our domain. The observer responsible for overall coding was also responsible for running the camera, noting the location and layout of the workspace, recording demographic information, and making general observations.

The coding schemes for interactions and events had to meet the following criteria: 1) capture critical events, 2) be efficient enough to use in real-time, 3) have a simple nomenclature so that the codes could be kept in memory, and 4) degrade gracefully as time pressure made real-time encoding difficult. We created the coding key (see Tables 2 & 3) from common events and interactions found in the literature seen in group work (e.g., [7]) and from the initial session where we observed the types of events and interactions that typically occurred in this domain.

Each coding sheet (on paper only) included a space for the interaction or event, the timecode correlated to the video tape timecode, and a comment. Since the interaction or event was the observable event, it was first written down with timecode and any comment following.

Interactions were coded for both activity between various participants (e.g., video editors, graphic artists, directors, assistant directions, clients) and between participants and artifacts such as the video on the monitor or the script they were reading from (see Table 2).

Figure 1. Generic ESDA Process [10] © Penelope M. Sanderson

Table 1. Attributes of Two Video Collection & Analysis Methods: Traditional Usability Testing vs. Observational Analysis

Traditional Usability Testing	Observational Analyses
Product/artifact driven	No a priori expectations - exploratory
Analysis of existing user interface	Domain and task analysis
Observation of a single user	Observation of multiple users, typically
Method for evolutionary product development	New product concept & system integration issues
AT:ST (analysis to sequence time): 1:1 to 3:1	AT:ST: 5:1 to 100:1
Lower cost	High cost
Broad analysis (e.g., major problem areas)	Detailed analysis (e.g., discourse analysis)
Tightly coupled to detailed design phase	Extrapolation of design concepts unclear
Non-UI experts can observe and analyze data	Expertise in observation and analysis required

These codes were driven by the video editing application area, but they may be applicable in other domains where people are using technology in collaboration. Specifically, this was an artistic, design task focused on temporal, graphical media that was being manipulated during the session. Thus, our coding scheme included interactions such as questioning statements, and gesturing actions, as well as events such as planning and individual work. Any coding scheme must be adapted to the domain studied.

Table 2. Codes for Modality and Type of Interaction

Person-Person Modalities	
Verbal	
C1-E1	C1 makes **statement** to E1
C1?E1	C1 asks E1 a **question**
C1-(*A*)E1	C1 makes a **statement about artifact** (*A*) to E1
C1-*	C1 makes a **statement to everyone**
C1-E1,E2	C1 makes a **statement to E1 and E2**
C1<>E1	**Conversation** between C1 and E1
Gestural	
C1-G-E1	C1 **gestures** to E1
C1-E(*A*)E1	C1 **points at** artifact (*A*) for E1
C1-G(*A*)E1	C1 **gestures over** artifact (*A*) for E1
Artifact Modalities	
C1--VE	C1 **interacts** with video editing equipment
C1-R-S	C1 **reads** Script
C1-W-S	C1 **writes** on Script
C1-L-S	C1 **looks** at Script
C1 = client, E1 = video editor	

In practice interactions were usually easier to detect than events because they required an observable action between people or between people and their environment. An example interaction between a client and a video editor follows:

A director points a finger at a transition that is a bit choppy in a cut between two clips and asks the editor to review that video segment again to determine how to alter it.

Typically interactions are easily observed and recorded. Although this example points out the difficulty of separating one interaction from another. Since the gesture is at the same time as the question, more than one atomic interaction code needs to be coded for the same video segment. Using our coding scheme, this example would be coded as follows:

C1-G(A)E1	director points at video cut
C1?E1	director asks editor to view again

Events, on the other hand, must be abstracted, to some degree, and were more difficult to encode on the fly. An example of an event in relation to the previous interaction is the "review video" event. Since we had little preconceived notions about the nature of the work we observed, our events changed somewhat during the observations and dramatically during analysis as described later. One instantiation of the event coding used during real-time coding is shown in Table 3. Another event coding table that was used during video tape coding as the model of the video editing events became apparent is presented in Table 4.

We believe the nature of observational data collection in an unknown domain will require coding of events to be fluid with many additions, qualifications, and revisions of the codes during observation. Our experience was that varying codes makes the analysis step more difficult, but better focused.

There are unresolved issues regarding the observation step of our process. Significant skill is required to do the coding, and this could be difficult for an engineer to learn quickly. To practice our skills for live sessions, our team coded in real-time while watching already recorded video tape as practice for the live session. The amount of detail that can be recorded must be adjusted according to how the codes will be used. We coded as much as possible, not knowing the correct level of detail necessary. With future observation/analysis cycles, we hope to better understand and predict the granularity of coding needed for specific analysis.

Finally, a computerized tool to assist in the coding would prove useful. This tool should allow the observers to quickly select events or interaction participants and types of modality and insert a timecode synchronized to the video camera's timecode. A real-time video logging system such as Marquee would be useful [13].

After the observations were completed, we prepared for the analysis step by moving the coding information into a database. We then used this database to drive our video tape recorder via a VISCA interface [2]. The database contained the coding information and a timecode association that allowed us to access the interesting portions of the video tape directly.

Table 3. Codes for Events used during observation

P	Planning
R	Review
IT	Individual task
CT	Important collaborative task
A	Approval
E	Equipment work (e.g., trouble with UI)
WE	Waiting for equipment (e.g., create video effect)
D	Dead time
I	Interruption
W	General waiting

Table 4. Codes for Events revised during model development

Review video	Overall approval
Trim video	Decide what's next
Find video	Plan for later
Approve video	Build context
Select video effect	Trim audio
Create video effect	Find audio
Order video	Laydown audio
Laydown video	Review audio
Choose between alternatives	Decide on audio effect
Overall review	Create audio effect

Analysis

*Model Selection Criteria***.** The goal of our analysis process is to produce a useful model of the application domain. The form the model can take is constrained by the needs of the consumers of the model, by the need to directly support design, and by the questions we want the model to answer.

Consumers of the Model. This model will be used by two different groups: a group knowledgeable in the tools and techniques of user interface design (ourselves) and product group engineers. The model must support the analysis activities we wish to perform. In addition, the model, or some transformation of the model, must quickly communicate key concepts to product groups.

Support Design. The model must directly support the design of an interface. For any interface element in the design, it should be possible to point back to a part of the domain described by the model that the UI element attempts to support. A prose description of the domain would not meet this criteria.

Questions the Model Should Answer. In order to drive design, the model needs to answer the following questions.

¥ Where is time spent in the current video post-production process? The answer to this question should reveal bottlenecks in the process that a new product can address. For example, if the majority of the time is spent in reviewing or trimming the video, separate functionality should explicitly support these parts of the process.

¥ What opportunities exist for new interface elements? One useful approach to conceiving of new interface elements is to reify domain objects in the interface. The model should highlight objects that are manipulated by collaborators and the referential objects of interactions between collaborators.

¥ What kind of telepresence infrastructure is necessary to support current forms of collaboration? An understanding of interaction modes (e.g. gesturing, speech) can drive the design of teleconferencing facilities.

An understanding of the content of interactions and where in the process they occur highlight possibilities for reifying interactions within the interface.

¥ What opportunities exist for parallelism? The model should record what each collaborator is doing at a given moment. Places where one collaborator is waiting for another are opportunities for making the process more parallel.

Script. Scripts are a modeling approach potentially meeting the requirements described above. Shank first developed scripts as a knowledge representation technique for capturing stereotypical events [11]. We did not use Shank's formal conceptual dependency notation, as it is too fine-grained for our purposes. The notions we borrowed from scripts are scenes, objects, and roles. A script divides an event into a series of named scenes. Scenes may branch, indicating multiple paths. Each scene has a list of the props (objects) and roles that appear in the scene.

The script notion has been useful for capturing segments of process. Each scene gives the process segment a name, includes a short natural language description of the segment, lists the objects that figure prominently, describes what each collaborator is doing during the scene, and describes the interaction mode and content of typical interactions that occur during the scene. Start and end timecodes of example video segments are also included in a scene. The flow of events in the process is captured by linking the scenes together.

Figure 2. Temporal structure of an observational data analysis meeting

The knowledge in a script can be handed off to a design team by recasting the script as a scenario. The scenario is a constructed stereotypical story based on a set of scenes. It is hoped that recasting the script in this way will yield a description of the domain that has low overhead for a design team to digest, and is able to capture the nature of the work.

This script approach meets the model criteria above. Each interface mechanism should point to a scene or set of scenes that it impacts. Objects that might be represented in the interface are listed in each scene. The description of interactions within each scene highlights both the interaction mode and the content. Descriptions of each collaborator's role in a scene provides information useful for increasing parallelism in the process. Finally, a transformation to a scenario exists that allows the information captured in the script to be transferred to design teams.

Structure of an Analysis Meeting. Following Jordan & Henderson's advice, our analysis meetings have been collaborative with frequent references to the video tape [5]. The temporal structure of these meetings is shown in Figure 2. Figure 2 is not meant to indicate the precise temporal structure of an analysis meeting but rather provide an example of a ÒtypicalÓ analysis cycle in such a meeting.

The three labeled bands along the abstraction axis indicate the three primary activities we engaged in during our analysis meetings. Moving opportunistically through these three levels of analysis has been an effective way to move through intermediate models on our way to the final construction of a script model. Before describing each of these activities, it is important to define what we mean by model. A model is a set of concepts (labels) organized by a set of relations. During analysis, we tried to label segments of video and relate these labeled segments in useful ways.

Coding. During coding, video segments were labeled with concepts from the model. This coding is distinct from the observational coding performed while we were videotaping sessions. The coding we performed during observation was real-time, using the interactions and events shown in Tables 2 and 3. The observational coding served to bootstrap our construction of a model during off-line analysis. As off-line analysis progressed, these model concepts changed dramatically.

Coding was characterized by almost continuous attention to the video tape. Different members of the team were generally concentrating on different aspects of the coding (e.g. events, objects, or transitions between events). Short comments were continuously exchanged during coding to verify that consensus had been reached. The tape was sometimes paused for a short period of time to discuss points of coding disagreement. If the disagreement was quickly resolved, coding continued. If the disagreement was more prolonged, analysis moved to model construction.

Model Construction. During model construction, we turned to discussing and modifying the model. Entry into this level of analysis occurred when one member of the team felt that none of the currently existing concepts mapped onto a segment of video, that several of the concepts had sufficiently vague definitions that a segment of video could not be unambiguously labeled, or that relationships between concepts needed to be created or modified. Discussion centered around determining whether an already existing concept could be applied to the segment or a new concept needed to be created. If it was decided that an existing concept applied, the definition of that concept (and possibly others) was modified so that the concept mapped unambiguously to the video segment. If a new concept was created, the definitions of other concepts were sometimes modified to make room in the model for the new concept. If discussion continued, we sometimes began questioning whether our entire modeling approach was appropriate. In this case, analysis moved to model evaluation.

Model Evaluation. During model evaluation, we discussed the desired properties of the model and whether the model as it was currently evolving satisfied those desired properties. Points discussed during this phase included the granularity of the model (is the number of model elements becoming to large?), the applicability of the model to design (how will the model help make design decisions?), and the amount of coding time the current model required (can we afford to take this long with our analysis?). This stage of analysis could be characterized as searching the space of models for a model with which to continue analysis. Because of the constraint that the model directly drive design decisions, discussion sometimes included searching the design space of a hypothetical system. An analyst might make a statement like "If we make distinction x in the model, and I'm able to code events y using this distinction, then we would have justification for a feature such as z." Purists may argue that including design space considerations in the analysis phase influenced our model of the domain. We hope that it did! We were not attempting to produce some sort of "value free" or "objective" model. The motivation for understanding the domain was to support the design of technological interventions.

Collaborative vs. Individual Analysis. Collaborative sessions have been an effective way to have discussions at the levels of model building and model evaluation. We have noticed some difficulty, however, with analysts wishing to operate at different scales of analysis. One analyst may want to rewind and watch the same segment of conversation multiple times because they are developing a detailed understanding of the interaction structure within a particular scene. Another analyst may be looking at the transition patterns between scenes, and want to move more quickly through the tape. If we stop to move slowly through the conversation, the analyst looking at transition information will loose their context. If we don't move slowly through the conversation, the analyst trying to capture the interaction details will become frustrated. Yet if we were to do all of the analysis individually, we would miss the fruitful discussions on the model building and model evaluation levels. As a compromise, a half-hour segment of tape was chosen for individual analysis. We then gathered as a group to discuss the results of our individual analysis and normalize our analysis concepts. Subsequent analysis sessions were collaborative until pressures grew to again explore divergent threads. Another half-hour segment was then chosen for individual analysis. It is still an open question of how to best combine the benefits of collaborative analysis with the benefits of letting different analysts work at different scales of analysis.

Transfer of Observational Analysis to Product Groups. To turn our observational data analysis technique into an engineering method, we would need to provide product groups with a general model architecture (such as scripts), and a set of guidelines for model construction and coding. By providing an architecture with guidelines for application, product groups would not have to engage in model evaluation. Even with model evaluation removed, observation analysis would still be a time consuming method. This may mean that to employ this methodology in an engineering product environment, the modeling will have to be done by specialists. This danger in this approach is in the critical hand off of the model and recommendations to the development team. It is an open question for us whether enough analysis guidelines can be stated in enough detail to reduce analysis time to a level that product teams can tolerate.

Collaborative Analysis History

Identification of Interesting Elements. In the first stage of analysis, we attempted to capture scenes on index cards. Using the codes from the real-time observation coding, we would cue to a location on the tape, watch it, and create an index card describing the scene. At this point, the scene was merely given a name. The idea was to first enumerate the scenes, and later detail them with roles, objects, etc. We found that at this point, however, it was difficult to restrict our notes on the index cards to scenes only. The index cards became a repository for "interesting things." Recording these categories on index cards facilitated sorting, exploring and changing the set of categories. As we watched each new portion of video tape, we could sort through the cards to decide whether this portion of tape fit any of the categories. If it did not, new categories could be created and old ones changed, discarded, or combined with others.

Product Group Presentation. A couple of weeks into analysis, we were asked to give a presentation of our research to a product group. We had to take a snapshot of our analysis effort to date and present it as a freestanding model. We sorted through the categories of events we had collected so far and came up with a core set of events. Now we had to temporally relate these events to complete the model.

All of us had the strong impression from our observations, reinforced during analysis, that the process has a self-similar hierarchical structure. For example, the "review" event happens at many different levels, all the way from a global review of the entire video piece (e.g., TV commercial) to the detailed review of a single video dissolve. We first tried to capture this hierarchical structure in a task hierarchy. This approach broke down for two reasons. First, the hierarchy we were dimly perceiving was not a hierarchy of control, as is typically captured in task hierarchies, but a hierarchy of scale. Second, it was becoming evident that the video post-production process is characterized by an opportunistic rather than a hierarchical process.

Next we tried to apply a network architecture such as that described by Olson [7]. In a network model, events are represented by nodes of a graph. The arcs represent transitions between events. Multiple weighted arcs can enter or leave a node, representing the weighted occurrence of sequences of events in the data. The network model seemed promising since we could represent the recurrence of an event (such as "review") at many places throughout the process without the constraint of a hierarchy. While this approach seemed promising, it was too complex to generate this model in the short time we had before the presentation. Further, the resulting model would have been too complex to present during a short presentation.

Finally, we settled on a procedural model (figure xx). We imposed a "reasonable" generic order on the events within the procedural model. We attempted to capture the flavor of the multi-scale occurrence of events by sprinkling the most common multi-scale events (review, decide, act, approve) throughout the process. While this model was successful in communicating with the product group, it doesn't meet our model criteria. The most useful aspect for our work of preparing for this presentation was that it forced us to explore a range of models and their features early in the analysis process.

Model Structure Chosen. After the presentation we continued to explore the network model. An extension of this model is what we are currently using in our analysis. Each node of the network is a scene rather than an atomic event. Scenes can be opened up to reveal a subnetwork of subscenes. This captures the multi-scale nature of some scenes.

Figure xx provides an example of a scene (Review Video). In Review Video, an editor and a director work together to judge a video segment against a set of criteria. The vagueness of the interactions in this scene is a function of the level of abstraction. Review is a fundamental activity which occurs frequently at many different scales throughout the video editing process. The large number of more concrete review scenes (the subtypes) are a consequence of this abstractness. Each of these more concrete review scenes provides a more detailed description of the interactions and artifacts involved.

The Model

Networks. An example of a network appears in figure xx. This network represents the work process captured from a half hour video tape of an editing session. The labeled boxes are scenes - distinct activities abstracted out of the work flow. By drawing such networks for different editing sessions, we were able to generate a convergent set of such scenes for the domain of video editing. The numbers on the links between scenes represent the number of transitions which occur between two scene types. When a large number of transitions are observed between two scenes or a group of scenes (as between Review to Generate Context and Build Story Context), this highlights an opportunity to provide explicit system support for the coupled activity in the interface

Scenes. Scenes consist of a description of the roles (different participants) participating in the scene, the typical interactions which take place between the participants during the scene, and a list of the artifacts which play a prominent role in the scene. The scenes are arranged in both an abstraction hierarchy and a partology (sub-part, superpart hierarchy). That is, some scenes capture abstract similarities shared by a group of related activities (abstraction hierarchy) while other scenes represent complex activities consisting of several smaller scenes which tend to occur in tightly coupled clusters in the networks (sub-part, super-part distinction)

Review Video

Definition:
Looking at a video segment to check it against criteria, to set context for collaboration, or determine worth or appropriateness.

Roles: Editor, Director.
Interactions:
Editor -> Equipment
Editor -> Script
D -> E: Query about status of task
Artifacts: Criteria (script, aesthetic, video in mind), Video segment.

Subtypes

1. Context Review -- Review of video segment to remind editor and client of what is there.
2. Design Rationale Review -- Review to give editor opportunity to explain editing decisions.
3. Review to Determine Use of Clip -- Review to determine whether a specific take was used earlier in the show.
4. Movement through timeline review -- Review through timeline to locate next work area.
5. Clip Evaluation Review -- Determining whether a clip will be useful for next edit.
6. Serial Review of 2 Takes for Decision -- Almost parallel review of two different takes for detection of differences and a decision about which parts to use.
7. Trim Review With Collaboration -- Client reviews video while editor is doing his own internal review.
8. Trim Review -- Short review of trim for editor to view trim work.
9. Effect Review -- Review video to determine if effect was created correctly for desired look.

Figure xx: The Review Video Scene

Design

The goal of identifying a model of the video editing process is to directly drive the design of a system to support this process. This section describes one system element and how it derives from the model.

The Script. As we developed the catalog of scenes in our model, we noted that the script plays an important role in many of these scenes. It is the primary artifact carrying the vision of the video to be created from pre-production, through production, to post-production. During post-production the script is used to note design rational (which takes were chosen for which segments and why), as a reference to determine which takes were digitized and why, as a planning tool to decide what must be accomplished during an editing session, and as a tool to set and regain story context (easy to loose when editing the details of, for example, a montage sequence). Many of the director, editor and client interactions centered around discussions of the script. It became clear that any editing interface to support collaborative editing at a distance must provide explicit support for manipulating and discussing the script. In fact, we feel that non-collaborative editing systems would have much to gain with such a metaphor as well. A storyboard of such a script interface is shown in figure xx.

This interface consists of two columns: a "Script" column and a "Production" column. The script column consists of some number of scenes (in the video sense, not to be confused with the scenes of our model). The interface for script manipulation should support all the features of a word processor plus additional structure to support script writing.

The production column contains representations of artifacts generated or used during production. This includes production notes, folders of video/audio/graphics clips, scheduling information, and status information.

The simple spatial relation provided by associating production artifacts with scenes in the script is very powerful. For example, the relations provided by the script can later be used for search and retrieval. Such user tasks as Find Video Based on Location in Script and Find Video Based on Timecode in Script (two identified scenes in our model) are transformed from unstructured and unaided search tasks to simple navigational tasks supported by the system. For example, if a an editor wants to retrieve all of the takes for a specific video scene, it only requires going to that scene in the script and opening up the folder of pointers to video takes.

The identification of the script interface as the primary metaphor organizing our proposed editing system followed directly from our model of the post-production process.

Organization and Distribution of the Analyses

A final, critical goal of this work is to effectively communicate what we have learned about this application domain to product teams within Tektronix. We have implemented system support on the fast-growing World-Wide Web (WWW) to facilitate reaching this goal [1]. We took the hypermedia infrastructure that the WWW provides and used it to produce a project-specific database to record our data analyses, as well as all of our other work associated with this project. The system provides four important features: 1) forces us to organize and formalize the products of our analysis; 2) provides a dynamic, rich, hypermedia infrastructure for recording our modeling activities; 3) provides a means for easy and attractive distribution of our analyses to interested product teams within Tektronix; and 4) supports our process for collaborative analysis as described in the previous section.

Figure 3 shows the information model for our hypermedia web. There are four main databases: research references, product evaluations, the data analysis base, and a UI mechanisms base. Each entry in each database is a WWW page with hypermedia links to other pages or multimedia objects such as graphics, video, or audio. Any page can contain links to any other page in any database, but in practice we have introduced more links within than across the main databases. Further, because of the distributed structure and uniform nature of the WWW, links can exist to reference resources anywhere on the WWW. For example, we have links in place to a Wide Area Information Search (WAIS) server that contains a bibliography of user interface research [9]. We have a project home page to allow easy and organized access into the entire structure, a number of indices, and a search engine to allow keyword-based search into one or more of the main databases.

To support rapidly adding to our hypermedia web we have created the following templates: research references, product evaluations, UI mechanisms, and multiple templates for the data models base (described below). We have also developed some conventions among ourselves for the use of the templates to assure that logical and visual consistency is conserved. For example, one convention alluded to earlier in the analysis context is that if we add a UI mechanism to the mechanisms database that we must provide a link from that mechanism to one or more of the data analysis pages. This conventions provides two benefits: 1) it helps ensure that design of UI elements is driven by the appropriate data analysis and 2) it is a means of easily and flexibly recording design rationale.

One important feature of the system is that it provides a very rich ability to represent the models from our data analyses. The script model described above, for example, is particularly well suited to being represented in hypertext. Our script template includes fields for objects and roles, as well as links to the scenes and sub-scripts that are part of the script. Each scene template has fields to allow a prose analysis of the scene, and hypertext links to other scenes to represent scene transitions. Ordering of scenes is naturally provided by the hypertext links, as well as looping of scenes, multiple branches between scenes and scene-subscene relationships. Links to research references, product evaluations, or UI mechanisms all may be directly embedded within the analysis as needed. Example cuts from the original observational video may be digitized, stored on disk, and embedded as a hypermedia link within a scene document. This hypermedia structure allows

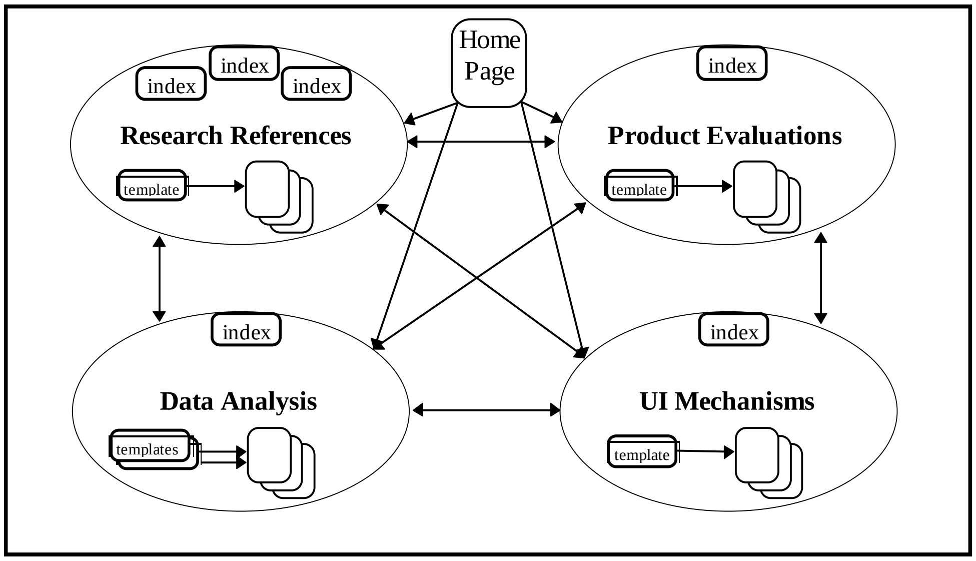

Figure 3. Information Model for Project Implemented as Hypermedia World-Wide-Web Database

us to easily ground the data analysis in actual video from our observation. In addition, the structure makes the analysis more entertaining and more easily comprehended by someone browsing through the database for information relevant to design.

A key aspect of the system is that it strongly supports our process for data analysis. Each member of the project team can read and write database entries, and can link entries as relationships are perceived either individually or consensually. This makes the web dynamic and provides a way to capture the relationships between our individually and collaboratively produced models as they evolve. Each template has a comments and links section that allows each of us to individually comment on the page contents, explain how it relates to other pages or recursively comment upon another colleagueÕs comments and have the dialog directly incorporated into the database. The comments may include prose and links to multimedia such as voice annotations, example video/audio clips from our observation, drawings or simulations of UI concepts, or other documents.

The entire database serves as a snapshot of our groupÕs current understanding of a domain, dynamically being updated as that understanding is enhanced through our collaborative data analysis, added references, new product awareness, or design ideas. This snapshot is then made immediately available to project team members on the platform of choice through the use of common WWW browsers such as Mosaic.

Evaluation

Evaluation of any new process is difficult. You do not have the luxury of being able to look back over time and see generations of successful products or the evolution and the large scale adoption of the method. There are, however, a number of valid perspectives from which one can extract early measures of success. These perspectives include the sponsoring agent, the directing manager, the early users of derivatives, the behavior of competitors, direct successful products, projections on to other problem spaces, and the ability to address criticism of other systems, as well as the evaluation by the research team.

This project went from a marginally tolerated exploratory investigation to a fully recognized and featured research project in six months. Top divisional management recognized and drew upon the expertise (recently developed) of the research team to discuss vision and strategy in this domain with key customers. These are indictors that the early results were seen as relevant and immediately useful.

The directive from the lab director was to search for engineering methods which could extract design data and enhance the front end of the customer-centered design methodology already in place. The method needed to allow HCI researchers to enter into an unfamiliar domain, observe it, characterize it, and extract product opportunities and design concepts. All this is to be done in the same amount of time typically devoted to product investigation prior to the concept development phase. The “first use” reported here included the development of the methods from scratch. The team went from approval of the project to identification of design concepts in four months and to the prototyping stage in six months. The research project was judged to be well within the needed time frame. With this experience it is expected that this team could approach a new domain and reach the prototyping stage in three months. The learning time and execution time for a totally new team is yet to be determined, but there is reason to be optimistic that this method can smoothly fit into an engineering design process.

The design teams who were the first recipients of the information agreed with the domain characterizations and implemented some of the design concepts that were indicated in the research. Unfortunately it is too early for us to be able to report on the successful implementation of an full system that was generated from this method.

Another means of validating design research is if it is confirmed by the work of others. In the commercial realm, this means that you begin to see the design concepts independently uncovered begin to appear in products and marketing visions in the marketplace. A number of the key ideas that were the result of this research were seen for the first time in products demonstrated at the NAB (National Association of Broadcasters) in 1995, four months after the finish of the concept development stage of this reported research.

Another way of assessing the success of a design process is to project how it would perform in a different environment. Would it catch major problems missed by other methods or identified by others as problems. In Transforming Work: Collaboration, Learning, and Design, Patricia Sachs [ ] points out the difficulties in re-engineering an organization or redesigning a system. She argues that if one were to take the common organizational view (explicit) which captures process flow (task analysis) one would miss the tacit (activity view) elements which lead to problems in use and costly work arounds. The specific domain reported involved the Trouble Ticketing System, a process for identifying problems, assigning responsibilities, and tracking solutions in a major communications system. The method reported in this paper,. if applied in this domain, would have captured both the organizational and the tacit elements of the work since our method focuses on the activities, the interactions, and the events and forms models of the work that include all of these elements.

Hughes, King, Rodden and Anderson point out in their 1994 CSCW paper, Moving Out from the Control Room: Ethnography in System Design [ ] that the two major challenges faces by these types of methods are to be able to respond to the time constraints (not be a 'prolonged activity') and to be able to frame results so that they may be used by system designers. Our method addresses both of these concerns as well as supporting the need for the researchers to be engaged in the domain and be doing their work from the perspective of evolving the work practice (not merely supporting, but enhancing the way work is done). It fits into the category of "Quick and dirty ethnography." I might be better named as "Quick and effective ethnography."

Our own assessment (we who conducted the research) is based on our collective experience of applying customer-centered design methods over period ranging from 1 to 22 years. This method shows great promise and was more successful in its first application than any of the other methods we had experience with. This was a challenging, exciting, demanding, and sometimes frustrating experience for our research team, but we felt that this was a successful effort and that we would seek opportunities to apply and improve the method in a new domain. Our team is confident that this method has the potential for, over time, driving research agendas as well as product development.

Summary

Even in the relatively crude, "first use," stage of this method there are very encouraging assessments from many of the stakeholders of the development process. This method has demonstrated the ability to identify product opportunities and design concepts that apply existing technologies in new applications. It has the potential for driving research and development of new technologies and product categories based on demonstrated domain opportunities. Clearly, for the researcher, this is a promising path to pursue both in the future refinement of the method and in the development of supporting methods and tools.

References

1. Berners-Lee, T., Cailliau, R., Groff, J.F., and Pollermann, B. (1992). World-Wide Web: The Information Universe. Electronic Networking: Research, Applications and Policy, 2:1. pp. 52-58.

X. Blomberg, J., Giacomi, J., Mosher, A., Swenton-Wall, P. (1993). Ethnographic Field Methods and Their Relation to Design. In Participatory Design: Principles and Practices. Schuler, D. & Namioka, A. (Eds). Lawrence Erlbaum Associates. Hillsdale, NJ.

2. CVideo (1992). *CVideo™: Macintosh¨ Video Annotation Software.* Envisionology. San Francisco: CA.

3. Grossmann, S., Lynch, G., & Stempski, M. (1992). Team approach improves user interfaces for instruments. *Electronic Design News (EDN).* June 4, 1992. pp. 129-134.

4. Heath, C. & Luff, P. (1992). Media space and communicative asymmetries: Preliminary observations of video-mediated interaction. *HCI.* 7. pp. 315-346.

x. Hughes, King, Rodden and Anderson (1994)

5. Jordan, B. & Henderson, A. (1993). Interaction Analysis: Foundations and Practice. From Human Factors and Ergonomics Society 37th Meeting Workshop: Exploratory Sequential Data Analysis (ESDA). Making Sense of Observational Data. HFES: Santa Monica: CA.

6. Knox, S., Bailey, W., & Lynch, G. (1989). Directed Dialogue Protocols: Verbal data for user interface design. Proceeding of *CHI'89*. pp. 283-287. New York: ACM.

7. Olson, G., Olson, J., Carter, M. & Storr¿sten, M. (1992). Small group design meetings: An analysis of collaboration. *HCI.* 7. pp. 347-374.

8. Palmiter, S., Lynch, G., Lewis, S., & Stempski, M. (1994). Breaking away from the conventional 'usability lab'. *Behaviour & Information Technology*. 13(1-2). pp. 128-131.

9. Perlman, G. (1991) The HCI Bibliography Project, ACM SIGCHI Bulletin, 23:3, pp 15-20.

x. Sachs, P.

10. Sanderson, P. & Fisher, C. (1993). Exploratory Sequential Data Analysis: Foundations. From Human Factors and Ergonomics Society 37th Meeting Workshop: Exploratory Sequential Data Analysis (ESDA). Making Sense of Observational Data. HFES: Santa Monica: CA.

11. Schank, R.C. & Abelson, R.P. (1977). *Scripts, Plans, Goals, and Understanding*. Hillsdale, NJ: Erlbaum.

12. Suchman, L. & Trigg, R. (1991). Understanding practice: Video as a medium for reflection and design. In J. Greenbaum & M. Kyng (Eds.), *Design at work*. Hillsdale, NJ:LEA.

13. Weber, K. & Poon, A. (1994). Marquee: A tool for real-time video logging. Proceeding of *CHI'94*. pp. 58-64. New York: ACM.