Designing representations for groupware by examining discourse

Alexander Feinman and Richard Alterman
Brandeis University Computer Science Dept., MS018
415 South Street, Waltham, MA 02454 USA
Email: afeinman@cs.brandeis.edu
Fax: 1-781-736-2741

Author Keywords
Analysis, computer-mediated communication, discourse analysis, ethnography, groupware, reference.

ACM Classification Keywords
H5.3. Information interfaces and presentation (e.g., HCI): Group and Organization Interfaces. *Computer-supported cooperative work, evaluation/methodology.*

ABSTRACT

Groupware requires careful design work to construct systems that help, rather than hinder, online coordination. Understanding the emergent work practice of participants in a joint activity is crucial to designing effective software. We have adapted ethnographic techniques to create two analysis methods. These methods can be used by an analyst to systematically examine the emergent work practice of participants and recommend appropriate representations for shared information.

One method tracks the references participants make in their discourse to give insight into how participants exchange and store information. This informs the design of appropriate representations and procedures for storing and exchanging information. In this paper, we show how this method provides a basis for designing mediating representations for a groupware system and predicts their use and misuse. We also show evidence that students were able to learn and apply these methods to analyze and redesign groupware systems they constructed.

INTRODUCTION

Computer-mediated collaboration has become ubiquitous. Remotely-taught educational courses, collaboration with work divisions in remote locations, and coordinating military personnel distributed across a net-centric battlefield are all domains where same-time / different-place [5] interactions can become difficult. Participants in different locations have access to different physical environments; hence, the methods they use for interacting are different than those used in face-to-face interaction. Procedures for referring to, pointing at, modifying, and reviewing objects, as well as gauging the focus and intent of other participants must necessarily be altered when participants interact online.

The goal of a groupware system is to support these altered procedures and allow participants to get their root task done efficiently. However, despite the best efforts of designers, groupware applications often end up interfering with or fundamentally altering the very work they are designed to support [7, 19]. It is crucial, but difficult, to provide a system which match the needs of the participants. As participants perform a joint activity, a work practice emerges which can be hard to foresee. Ideally, software would be built to match this emergent practice [12, 20, 26].

One way to understand the emergent work practice is to take the same ethnographic techniques used to examine situated activity of participants in face-to-face activity and adapt them to online collaboration. We have adapted concepts from both conversation and discourse analysis to create methods that investigate the interaction at a conversational level. These methods organize ethnographic observations and generate concrete conclusions about the emergent work practice of participants: what sorts of information are exchanged, what areas of the interaction are problematic, and where the system needs to be modified to reduce coordination effort.

This paper will start with a brief introduction to one of our analytic methods in the context of an experimental domain. After this, we present an experiment which uses our methods to redesign a groupware system, and show that it can be used to predict how different representation designs will affect the work practice. We then present evidence that these methods can be taught and applied by comparing the results of three years of student projects in an HCI class.

BACKGROUND

Our methods grew out of the application of techniques from discourse analysis to the distributed cognition paradigm. Distributed cognition investigates interaction by examining the representations, both internal and external, available to participants [12–14, 21]. By enumerating these representations and determining the procedures used to store, update, and retrieve the information they contain, as well as transcribe information between representations, an analyst can gain a good understanding of the interactional work participants are performing. By identifying this work, an analyst can see how participants use the tools in their environment to complete tasks and to interact with others, and can gain an understanding of what sorts of tools should be provided to improve performance.

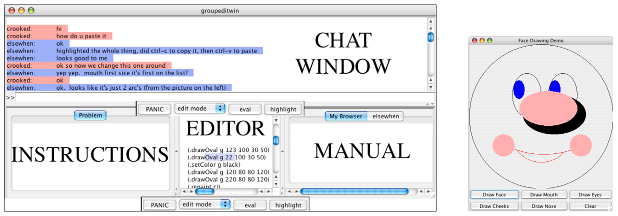
However, distributed cognition does not specify a principled way to draw quantitative conclusions about observations. To address this shortcoming, we have adapted techniques from conversation and discourse analysis. Conversation analysis [22–24] and related techniques have been used to examine the minutiae of interaction at the conversation level. It identifies the specific devices, such as conversational openings or adjacency pairs, that participants use to organize their talk. Characteristics such as turn-taking, speaker choice and speech act type have been used to identify breakdowns in coordination by highlighting departures from a standard model of interaction [8]. Examination of the duration and type of conversational utterances has also been used as a way of determining the impact of alternate representations on conversation [15].

The methods we have devised use features of the online discourse generated by participants to reveal the representation work that they perform. Our approach yielded two methods. The first of these is *recurrence analysis*; building on previous work [18, 26], it looks at recurring conversation about coordination, recurring errors in coordination, and secondary structure in the discourse. The other method *referential structure analysis*, is concerned with determining what types of information participants discuss, and how they communicate, refer to, and store task information. This paper focuses on the latter of these methods, and on how to use it to convert observations into design recommendations.

METHODS

To illustrate the methods we present an example taken from one of our experimental domains: the Group Homework Tool [16]. This project studied the educational impact of having students program in pairs versus solo programming. Two students are situated at two remote computers, and asked to complete a coding assignment together using a chat tool, a shared editor, and a pair of shared web browsers. Before and after the assignment, the students are individually tested on their programming skills; the results are then compared to examine knowledge transfer during pairs programming.

A screenshot of the Group Homework Tool is shown in Figure 1. In the center is a shared editor; above it is a chat window. On either side are shared browsers to view the instructions and the reference manual for the programming language. All communication and actions are recorded for later analysis. The assignment in this case was to draw a cartoon face, with various specified features such as a nose, mouth, and so forth. A sample result is shown on the right side of Figure 1. Figure 2 shows a portion of the chat transcript from near the beginning of the session that generated the face. Note that all dialogue is copied exactly as it was typed, with elisions, misspellings, and the like retained intact; necessary interpolations of the dialogue are indicated with square brackets.



```
crooked:     hi
crooked:     how do u paste it
elsewhen:    ok
elsewhen:    highlighted the whole thing, did ctrl-c to copy it, then ctrl-v to paste
elsewhen:    looks good to me
crooked:     ok so now we change this one around
elsewhen:    yep yep.  mouth first sice it's first on the list?
crooked:     ok
elsewhen:    ok.  looks like it's just 2 arc's (from the picture on the left)
```

```
(.drawOval g 123 100 30 50)
(.drawOval g 22 100 30 50)
(.setColor g black)
(.drawOval g 120 80 80 120)
(.drawOval g 220 80 80 120)
(.repaint c))
```

Figure 1. Interface for the Group Homework Tool experiment; to the right, the face produced by the code.

		Discourse	*mouth*	*plan*
7	B:	yep yep. $\underline{\text{mouth}}_a$ $\underline{\text{first}}_b$ sice $\underline{\text{it's}}_c$ first on the list?	a, c	b
8	A:	$\underline{\text{ok}}_a$		a
9	B:	$\underline{\text{ok}}_a$. looks like $\underline{\text{it's}}_b$ just 2 arc's (from the picture on the left)	b	a
10	B:	although I'm not sure what the parameters for .drawArc are.....		
11	A:	how will we be abe to see if $\underline{\text{its}}_a$ correct	a	
12	B:	well, $\underline{\text{[it]}}_a$ doesn't have to be 100% correct i'm guessing..$\underline{\text{[it]}}_b$ just has to look similar	a, b	
13	A:	ok		
14	B:	so we just use the eval. button and pray $\underline{\text{it}}_a$ looks ok :)	a	

Figure 2. Analyzing GHT data with referential structure analysis.

In this example, a pair of students have completed their pretest and are beginning their problem-solving session. The two students, who have not met before, are of disparate skill levels, and as a result user B ends up tutoring user A for most of the session. At the start of this excerpt, user B has copied and pasted some sample code from the instructions into the shared editor. The two users begin by discussing what part of the multi-part assignment to do first, and then move on to discuss implementation details. The students are provided with some sample code, which they use as a starting point. The conversation quickly turns to a discussion of how to address the first portion of the assignment – using the `drawArc` function to draw a cartoon mouth, and using the 'eval' button to evaluate the code to fine-tune the appearance of the mouth.

Analyzing the dialogue with referential structure analysis

Referential structure analysis grew out of an expansion of techniques for extracting discourse structure [17] by incorporating concepts from literature on coherence and reference resolution [9–11], as well as literature on the formulation of and grounding of references [2, 3]. The method involves tagging references in the discourse and combining them into coreference chains. By assigning types to the referents pointed at by these references, and computing statistics about these referents, we make visible the ways participants handle different types of information. Although grounding anaphoric references and forming reference chains can be problematic in the general case [27], we have achieved good results by restricting the set of referents under consideration and aiming for investigative rather than comprehensive results.

The aim of the method is to discover what sorts of things participants in a joint activity spend their time talking about, how much they talk about them, and for how long. For example, an analyst might investigate how frequently participants refer to some domain object, or how long they spend discussing a plan for action. To do this, the analyst tags the references made by participants and consolidates them according to the referent they refer to, classifies the resultant referents into types, and computes various parameters about each referent identified in this fashion.

In Figure 2, we show the results of tracking two referents through sample dialogue from the GHT domain. There are many possibilities for reference: code constructs, elements of the instructions, plans for action, the shape of the desired output, division of labor, and so forth. For clarity, we have pulled out only two referents from the dialogue, and marked each reference with a unique subscript; references are sorted into separate columns depending on which referent they point at. We note the type and every reference to each referent. From this we can compute two fundamental measures: lifetime of relevance, defined as the duration (in utterances) between the first and last mention of the referent, inclusive; and the number of references to that referent over its entire lifetime. These measures reveal certain important aspects of how participants make use of that particular type of information.

The first referent examined is the "mouth" referent, referring to a portion of the face which the code is meant to produce. References to it are collected in the second column from the right. On line 7, B refers to it by name ("mouth"), followed by an anaphoric reference ("it's"). On line 9, B refers to the mouth — in this case, the picture of the mouth provided in the instructions. Discussion on lines 11 and 12 refers to the mouth a number of times, including by means of elided pronouns; and at the end of the dialog on line 14, B refers to the mouth once more. From this, we can see that the mouth referent remains relevant for some time — in this segment, the first reference to it is on line 7, and the final reference is on line 14. (Conversation about it continues past this brief segment of dialogue, but we are restricting analysis to this segment for didactic purposes.) For this segment of discourse, it has a lifetime of relevance to the participant of seven utterances (inclusive), and is referred to seven times in five separate utterances during that lifetime.

Participants also discuss a plan for drawing the mouth at the start of the extract. This "plan" referent is referred to differently than the "mouth" referent. References to this plan referent are noted in the right-most column of Figure 2. On line 7, B proposes the plan ("mouth first [. . .] ?"); on line 8, A accepts ("ok"), and on line 9, B acknowledges acceptance ("ok"). This constitutes the entire conversation about this plan. In contrast to the mouth referent, the plan for ordering tasks is relevant for three utterances (lines 7–9), but after this is never discussed again. Although the plans continues to be a topic for discussion — lines 10–14 are primarily concerned with how to carry out the plan — the participants have committed to it and do not refer to the plan itself again. This particular plan referent has a short lifetime of relevance (three utterances), and is mentioned three times over that span.

This contrast in referential structure — a long lifetime versus a short one, and a period of frequent reference versus a lower-density but more continuous pattern of reference — is the sort of observation the method is designed to find. By examining these patterns of reference to referents, we can gauge how participants are exchanging information, and determine how best to mediate that exchange.

USING THE METHODS TO INFORM REDESIGN

Our experimental methodology for investigating a domain begins with examining existing work practice. For a real-world task, this would be performed by gathering ethnographic information through observation of workers at work. However, these techniques can be somewhat limited; it is difficult to record a complete picture of the interaction. Even high-fidelity techniques such as video recording may lose a fair amount of interaction detail due to limitations of the recording and playback technology. However, with computer-mediated, different-place coordination, it is possible to use the computer itself to record the interaction. Because all interaction is mediated by the computer, constructing systems that record and play back what users do makes the entire interaction available for analysis. To investigate a domain, we construct a basic system for an initial study, have users perform their tasks on it, and analyze the recorded interaction. From this, we design and construct an improved system, and construct representations that are suitable for supporting the observed work practice.

The business travel domain

We devised a two-stage experiment to test the utility of methods for suggesting new representations and predicting the impact they will have on the emergent work practice. Specifically, we wanted to show that, using referential structure analysis, an analyst could predict which representations for information would be adopted successfully by users, and which would not be.

In keeping with our experimental methodology, we conducted a small study to gather initial data, and used this data to design a pair of domain-specific groupware systems. The first system, the "matching" system, was designed with a set of representations that matched the emergent work practice of the pilot study users, and were predicted to support the ways in which they shared information. In contrast, the "non-matching" system was designed with representations

Adam’s Mark Hotel $129 x 2 = $260 + tax
(google: hotels dallas)
Food:
Tues Lunch: near convention center ($30 each, $60 total)
Tues Dinner: bobssteakandchop.com (expected $160 total)
Tues Entertainment: random show 2tix x $45 = $90
Weds Breakfast: at hotel ($30 each, $60 total)
Weds Lunch: at Park: barbeque, $40
Sandy Lake Park – minigolf, $2 entry, $2/game: $12
- - - - - -
$682

Figure 3. Itinerary produced by a test subject in the Business Travel study.

which were very similar to those in the Matching system but did not match the work practice of users. Preliminary results from this experiment, discussed below, show that the methods used were successfully able to predict representation use, and give specific explanations of why users did and did not use the representations given.

The domain of business travel was chosen for its general familiarity among users. Specifics of the task were determined by conducting a brief ethnographic study of the methods used by real-world users to plan business trips. Subjects were given instructions such as the following:

You and your partner have 30 minutes to plan a business trip to the Dallas/Fort Worth area. You are scheduled to arrive at DFW Airport at 7:44 am on Tuesday, Sept 20th, and depart from there on Wednesday night at 4:17 pm. A rental car is waiting for you. You must spend 9 to 5 on Tuesday at the Dallas Convention Center. Stay within a $500 total budget. You should produce a detailed itinerary with times and budget to present to your support staff. Your tasks:

1. *Find a hotel.*
2. *Find places to eat meals.*
3. *Find entertainment for Tuesday evening.*
4. *Find a place to play golf on Wednesday.*

A small study was conducted (n=4) to gather the required data to design the systems. Pairs of subjects were asked to create an itinerary, including budgeting, for a two or three day business trip over the course of a 30-minute problem solving session. Subjects were given a private text editor, a chat client, and a web browser, and trained in the domain before being asked to plan a trip. Dyads generally nominated one member to construct the requested itinerary in the text editor; a sample appears in Figure 3.

Analyzing the base group data

Subjects reported some frustration with the task – organizing information was difficult, as was maintaining awareness of the actions of the other user. Despite an attempt to enforce the 30 minute deadline, no groups were able to complete the problem in under 45 minutes, with 50 minutes being the average time required.

Type	Freq	Refs	% of Refs	Lifetime	Density
Instance	31%	4.7	38%	12%	66%
Task	17%	4.7	21%	43%	24%
Plan	17%	1.9	8%	3%	80%
URL	11%	1.2	4%	1%	90%
Repair	8%	3.4	7%	2%	98%

Table 1. Referential structure data from the Business Travel domain.

Chat data from the base group was tagged using referential structure analysis. As a part of this process, we identified a number of new referent types: tasks, instances of tasks, and URLs. Some of the more generic types found in previous analyses (plans, repairs) also appeared in this data set. The most frequently-occuring referent types found as a result of this analysis are shown in Table 1.

Tasks are the generic tasks that users discussed. Finding a hotel, looking for entertainment, and calculating the budget were all considered tasks. **Instances** are specific places or events which the users find to fulfill a particular task; for example, the “Adams Mark Hotel”, a hotel found by multiple groups, was an instance which satisfies the task of finding a hotel. **URLs** are just that – references to specific web pages made by users. While there were other referent types found, these three new types, together with domain-independent referent types **plans** and **repairs**, accounted for a vast majority of all referents found (85%), and so we focused our attention on them.

For each type of referent we calculated a variety of measures (see Table 1): the *frequency* of the type (number of referents of that type, divided by the total number of referents); the average number of *references* to each referent of that type; the *percent of all references* which were to referents of this type; the average *lifetime of referents* of this type (number of lines of chat between first and last reference to a referent, divided by the total length of the session it appeared in); and *average density* (what percent of lines over a referent’s lifetime contain a reference to it).

We also calculated the concurrence of each referent type, shown in Table 2. This is the average expected number of referents of that type that are relevant at any one time. This was computed by noting how many referents of a type are relevant during each line of dialog (i.e., have both a reference before or on, and on or after, that line). As an average did not seem to adequately express the characteristics, we computed a number of additional statistics: the mode of the number of concurrent referents, and the maximum number of referents that were relevant at once, for each type.

Drawing conclusions

We shall discuss the utility of these measures as they apply to each type of information. By examining the statistics shown above we were able to come up with a desired set of features for representations crafted to match the pattern of interaction surrounding observed referent types. Representations were

Type	Mode	Max	Average
Instance	1–3	3–6	2.34
Task	3–6	4–6	3.24
Plan	0	1–2	0.19
URL	0	0–2	0.55
Repair	0	1	0.07

Table 2. Number of concurrent referents for each type.

therefore designed with features that matched the observed properties of information use, summarized in Table 3.

Tasks have the longest lifetime, at about 43% of a problem session. For such a long-lived referent type, they have a surprisingly high average density (24%); hence, it is unsurprising that their average concurrence is also high (3.24). This means that in general users are working on three or four tasks at any one time, with some spikes to six. This long period of relevance, coupled with the number of simultaneously relevant referents, clearly indicates the need for a persistent, shared representation which allows users to enter at least six and preferably closer to ten items.

Closer examination of how users talked about task referents revealed properties of tasks. Users initially spent some time discussing assignment of the tasks (e.g., "You find a hotel — I'll work on food."). However, after this brief discussion, references to tasks only occurred in the context of instances: nominating instances for a task ("How about the Marriott?"), or querying or reporting on the status of the task ("Do we have a hotel room yet?"). In other words, generic tasks had only a short lifetime outside their attachment to instances. This negotiation over division of labor can therefore be handled in chat, and the remaining activity about tasks can be combined into the representation for instances, which we will discuss next.

Instances were the most talked-about referent — accounting for over one third of references — and of reasonably long lifetime. Groups generally had between one and three instances relevant at any one time; one group had six relevant for a fairly substantial period. However, the long tail at beginning and end of sessions, where most instances were not relevant, reduced the average number of relevant instances to just over two. From this, we determined that a representation meant to store instances should allow at least six items, and probably more, to be stored at once, even though in general only a third of the representation will be in active use.

Users gave short-hand names to instances, and struggled with conversationally pointing at instances, especially when multiple tasks were relevant; hence, the representation should provide a naming field, to allow canonical references to items. Likewise, users spent a certain amount of time discussing the cost of an instance, and more time calculating the total expenditure (as required by the task specification). A representation for these therefore should include a way for users to input cost information in a structured fashion, so that the system can automatically calculate totals; this will

Type	Observed properties	Representation features
Task	Long-lived	Persistent, Nameable
	Frequent references	Dedicated screen area
	Coupled with Instances	Merge into Instance rep
Instance	Medium-length	Persistent, Nameable
	Frequent references	Dedicated screen area
	Moderate concurrence	Show multiple items
	"Budget" property	Structured storage for this
Plans	Short-lived	Ephemeral representation
	Almost no concurrence	No need for multiple items
	Few mentions	No need for naming
URLs	Short-lived	Ephemeral representation
	Very few references	Ephemeral representation
	Coupled with Instances	Treat as Instance property
Repairs	Short-lived	Ephemeral representation
	Very high density	Store in negotiable medium

Table 3. Mapping referent type properties to desired features of new representations.

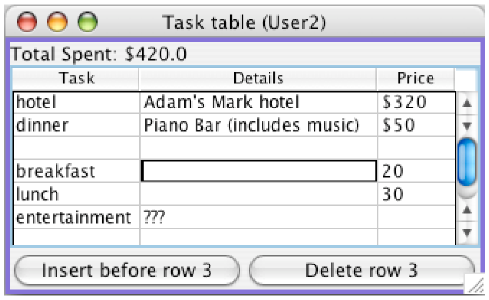
reduce cognitive workload, reduce conversation, and reduce incidence of error by offloading budget computations.

Plans occurred fairly frequently. However, as had been seen in previous domains, they had a very short lifetime (averaging 3% of a log file, or 3.6 lines of chat). Reference patterns were split; about half the time, a plan was proposed or reported once and never mentioned again. The other half of the time the plan was referred to a handful of times, indicating some negotiation. Only in a few cases did a plan continue to be relevant for more than five lines of chat. Plans almost never overlapped – in one case, a group discussed one plan while hashing out another, but otherwise only one plan at most was relevant at a time. Coupled with a high density — 80% — these statistics led to the conclusion that there was no need for a persistent, shared representation for plans: chat, with its ability to be used as both a tool for announcement and for negotiation of plans, was the best medium.

URLs were used by some groups to refer to web pages. These generally had one reference, though a few had two or three. These were always exchanged in the context of an instance; either a detail page for a previously-discussed instance, or as a way to begin negotiation over a newly discovered instance. As a result, these referents can be folded into the representation for instances, as an extra property. For the purposes of this experiment we decided not to implement this, instead focusing on providing representations for plans versus tasks.

Repairs were short-lived, and completely dominated conversation when they occurred. This is in keeping with findings for other domains. As in these other domains, we determined that chat is probably the best representation for these, as it allows pure, focused negotiation and repair of common ground.

Designing matching representations


Task table (User2)

Total Spent: $420.0

Task	Details	Price
hotel	Adam's Mark hotel	$320
dinner	Piano Bar (includes music)	$50
breakfast		20
lunch		30
entertainment	???	

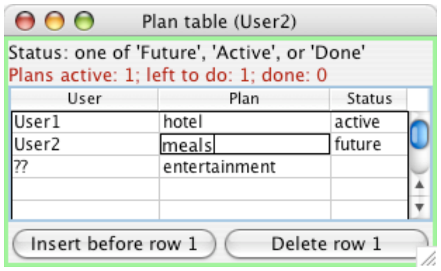
Figure 4. The Task Table, a representation designed to match observed work practice.

The goal of the base group experiment was to provide data to design two systems: one with representations that matched the emergent work practice, and one that conflicted with it. Examination of the data indicated that was crucial to provide an effective representation for instances and tasks, as together they accounted for well over half of all references. Based on the features predicted by the properties of the data — as mapped out in Table 3 — we designed a persistent grid representation for listing tasks and instances. This matched well with the secondary structure users created in their private text editors; all users listed itinerary data in a format similar to that seen in Figure 3. This led us to create a new representation for storing tasks and instances: the Task Table, shown in Figure 4.

The table consists of three free-form text columns. The "Task" column stores Task referents; the "Details" column gives users a place to put Instance information. The "Price" column is separated out to encourage users to enter information in a structured fashion, in this case, prices in dollars and cents. By using this structure, the system is able to provide intelligent support: prices are automatically summed up, the total is shown at the top of the window and compared to the total budget for the trip, with the text turning red if the budget is currently exceeded.

Designing non-matching representations
In contrast, the goal of designing the mismatched system was to predict representations that do not match how the analysis predicts information will be used. To this end we chose to encode plans in a persistent representation similar to the one devised for Tasks and Instances: the Plan Table is shown in Figure 5. As noted in the above analysis, plan referents tend to have a very short lifetime, with few references. This predicts that the work that participants expend to transcribe plans into the Plan Table would be wasted work; they should not need to store this information in a persistent fashion.

The first two columns give users space to note the person responsible for the plan, and a name for the plan. To avoid having users store instance details in this representation, space in the Task column was restricted. Users were instructed to put "Future", "Active", or "Done" in the third column, Status. As with budgeting in the Task Table, this structure


Plan table (User2)

Status: one of 'Future', 'Active', or 'Done'
Plans active: 1; left to do: 1; done: 0

User	Plan	Status
User1	hotel	active
User2	meals	future
??	entertainment	

Figure 5. The Plan Table, a representation designed to conflict with observed work practice.

allows the system to provide some intelligent support: the system automatically tracks the number of incomplete tasks, the tally is displayed at the top of the window, and is highlighted in red if there are still tasks pending. While this feature is of some limited utility, asking users to make use of the Status column in this way had the side effect of increasing the number of times users had to access a putative plan. To fully utilize the feature, users would have to 'refer' to a plan (by updating the row in the shared table) at least three times, well above the observed average number of references for plan referents.

As an aside, the plan table does provide storage for implicit references to tasks, as it allows entry of plans to perform tasks. It is a fairly good representation for task information; however, as we shall see, this did not provide enough support to overcome the mismatch between users' work practice and how the representation worked, and instead was subverted by users to store instance information attached to the tasks.

Predictions
Our prediction was that users would be happier and more efficient when using the matching system. Since the design data showed that users needed to talk about tasks and instances, the natural representation provided by the Task Table would be used as designed, and would help users complete their task faster and more easily. Users might also fold a representation for URLs into the Task Table, given that some groups used them to talk about instances.

Conversely, we expected users to resist using the Plan Table. The data indicated that planning was really best matched with the chat window. If users used the Plan Table at all, our predictions were that they would fail to update the Status field, or perhaps subvert the representation to store the data they needed to store persistently: instance information. As we shall see below, these predictions were validated by our data.

Experimental results
To test our systems we organized a small-scale experiment (n=4). A larger-scale trial is underway currently to improve the strength of results. Dyads were trained in the domain, and then given four problems to solve; two with the 'matching' system, and two with the 'non-matching' system. Half the dyads were exposed to the 'matching' system first; the

Measure	Observed change
Problems finished on time	‘Non-matching’ groups finished 33% fewer problems ($p < 0.1$)
Lines of chat	‘Non-matching’ groups generated 35% more chat

Table 4. Objective results comparing ‘non-matching’ system to ‘matching’ system.

Question	Matching	Non-matching
Reaction to system (1=hated it, 7=loved it)	4.9	2.9 $p < 0.01$
Ease of Use (1=difficult, 7=easy)	5.6	3.1 $p < 0.01$
Group coordination (1=poor, 7=excellent)	5.5	5.0 not significant
Group performance (1=poor, 7=excellent)	5.0	4.9 not significant

Table 5. Survey results from the initial BT study.

other half used the ‘non-matching’ system first, to counter-balance ordering effects.

We used a number of objective measures, coupled with user opinion, to evaluate the new design. These are shown in Table 4 and Table 5. The first objective measure was the length of time it took users to complete a problem. Allowing users to perform their tasks faster, all other things being equal, is a good sign of an improved system design. We found that subjects were able to finish problems on time more often when using the ‘matching’ system than when using the ‘non-matching’ system (86% complete vs. 57% complete — a 33% difference in completed problem rate). This was a good indication that the ‘matching’ system allowed users to perform their task more efficiently.

Another measure is lines of chat. Past work has shown that coordination work can be reduced by providing structured representations that match the way users communicate, transcribe, and store information. One clear indicator that this is occurring is a reduction in the overall amount of information sent via unstructured representations such as chat. Hence, by comparing the *chat output* of the initial system, where users are forced to coordinate strictly using chat, to the chat output in the new systems, where users have access to alternate representations, gives the experimenter a way to calculate how much information is being communicated via these alternate representations, a sign of their level of utility to users.

Our data showed that users chatted about 35% more when using the “non-matching” systems, a statistically significant increase. This was a good indication that coordination work that had moved to the Task Table in the ‘matching’ system was still being done in the chat in the ‘non-matching’ system. As noted previously, this allows an analyst a rough measure of the quantity of coordination work that has been shifted to the new representation.

User feedback and opinion were also used to gauge acceptability of the design. Designs which match the emergent work practice should be adopted with less resistance by users: they do not conflict with users’ expectations about how to perform their tasks, and they allow users to coordinate more effectively. The four measures presented in Table 4 were gathered via an exit survey, which also provided some free-form opportunities for feedback. As can be seen, users clearly indicated a preference for the ‘matching’ system, rating it a 4.9 vs. 2.9 (on a scale of 1 to 7). Likewise, they found it significantly easier to use — 5.6 vs. 3.1. Notable, however, was the lack of difference between feelings of group coordination and performance; despite objective measures to the contrary, groups generally felt their performance with both systems was above average.

Verifying predictions

The data matched our predictions fairly well. Users preferred the ‘matching’ system, were more efficient using it, and needed to resort to chat less. On the other side, users mostly disliked and complained the Plan Table, although one group found it somewhat useful and praised it in the exit survey. Users generally agreed that there was a greater need to communicate the information stored in the Task Table: “Task was much easier because it provided its own concrete space to enter the completed plans themselves, rather than just whether or not the plan was done.” Another user wrote, “The ‘plan’ system was confusing and frustrating because there was a way to record process but no way to record output.” One summed it up this way: “TASK is much easier[...] All planning really happens over chat.”

The matching representations were primarily used the way they were designed to be used, with some exceptions. For example, users shoehorned URL referents into the representation for instances, and then complained about the lack of proper support for URLs. As expected, the users chafed when using the non-matching representations, and in some cases forced them to fit their own needs. While this was not unexpected, the lengths that users would go to in order to store needed information were surprising. One group started the experiments using the ‘non-matching’ system and promptly began storing instance information in the “Plan” column of the Plan Table. This was despite training, and the use of design techniques to discourage this behavior (column width was harshly restricted to encourage storage of short information). Likewise, one group started storing URLs in the “Who” column. These users clearly saw a need to be communicating instance information, and created their own structure within the tools given to achieve their goals.

This data showed that we were able to use ethnographic observations of pilot study data in our design of two groupware systems, and predict the resulting usage of these systems according to principled analysis of those observations. Experimental subjects used the representations in the fashion that we predicted, including the re-tasking of representations to store critical task information in fashions that conflicted with their basic design. This showed the higher priority for the user of completing the task as opposed to making use of the system in its designed fashion, and highlighted a recurring design difficulty in groupware systems — mismatch of representation feature to task-driven information requirements — that these methods can help alleviate.

GENERAL APPLICABILITY OF THE METHOD

To establish the general utility of our methods, we performed experiments to demonstrate these three important qualities of the methods:

1. The methods can be taught to other analysts.
2. Students can apply these methods to redesign groupware.
3. Different analysts draw similar conclusions from the same data.

Teaching the methods

In the Fall of 2003, we performed an experiment involving teaching our analysis methods to a class composed of twenty-one Master's students and upper-level undergraduates. For the class project, the students worked into groups of two to four; each group created problems for pairs of subjects to solve cooperatively using the GrewpTool, a groupware framework similar to GHT. Students were asked to submit an initial design based on a survey of their available user population; topics ranged from "plan a 5-night vacation to Boston" to "the wedding dinner planner" to "create a web page describing the culture of a nation." After constructing a prototype of the system, students recruited three or four pairs of subjects, trained them in use of the system, and generated about 10 total hours of usage data. From this set of data the students were asked to select a single transcript and apply the methods presented in this paper to analyze the interaction.

Applying the methods

Students were given three weeks to generate and submit designs for new representations to improve user performance in their particular domain, with the requirement that these new designs be properly motivated using the analysis techniques discussed in class. Most groups were able to successfully apply our methods to suggest interesting redesign possibilities for their systems. Student groups that submitted redesigns were able to successfully motivate that redesign using one or both of the analytic methods taught in class.

Recurrence analysis

The first method we taught to the Fall 2003 class, *recurrence analysis*, is covered in more detail elsewhere [1, 6]; we will summarize it here. Recurrence analysis is a lens that can be applied during observation of a domain. It builds on previous work which examines ethnographic data for recurring indications of difficulty [26]. Identification of problem areas using these indicators allows an analyst to focus redesign efforts. The analyst notes interactions of three particular kinds as indications of problems in coordination:

- Recurrent communication about coordination: Situations where participants must repeatedly discuss their coordination to perform a joint activity. An indication that current procedures and tools for coordination may be insufficient.
- Recurrent errors of coordination: Situations where participants repeatedly commit errors. This is a good indication that the process is too difficult to perform correctly with the tools available.
- Creation of secondary structure: Conversational or procedural mechanisms devised and employed by the participants to help them coordinate their actions. It is a clear indication that the existing structure is insufficient.

All ten student groups in the class were able to identify recurring communication about coordination in the data, and used it to justify redesign. The recurring situations identified centered around the heart of the interaction in each case. For example, in a group that created a "wedding planner" system, the students noted users spent a great deal of time discussing seating arrangements. This focused their later referential structure analysis, and led them to create representations for arranging seating.

Almost all groups used the appearance of recurring errors as design justifications. For example, the subjects in one group were asked to plan a road trip from Boston to Los Angeles. They often made errors related to problems with attention; that is, one user would enter something into the shared text area, but the other user would fail to notice, and instead duplicate the efforts of the first user. As a result the designers proposed a representation that would allow users to keep track of what task each user was working on.

The appearance of secondary structure, the final indicator used in recurrence analysis, was less frequently utilized by the students — only two groups justified their redesign based on the appearance of such structure. The structure found by the students is nevertheless compelling. For example, in the "Boston trip" group, one of the subjects ended up filling the shared text editor pane with a highly-formatted itinerary. The subjects felt the need to create a shared representation to organize their activity; however, the tools at their disposal were minimal — only a shared text editor — and so they were unable to generate a truly effective representation. The redesign for this domain addressed this and other problems by including a tabular shared itinerary representation similar to the design of the Task Table.

Using referential structure analysis for redesign

About half of the student groups were able to further refine these design ideas with referential structure analysis. These groups used the regimen of identifying new referent types as a way to discover the information that participants discussed most frequently in their domain. They produced designs that incorporated shared, structured external representations for these kinds of information. These new referent types and new representations are summarized in Table 6.

These groups were able to focus their attention on the more important referent types, and so were able to design systems

Project domain	New types	New representations
Class web page	webpage	Browser history
Boston Trip	event	To-do list
	location	Itinerary
	price	Budget calculator
Themed web page	requirement	Requirement list
	topic	Topic list
Wedding dinner	constraint	Seating Chart
	food	Menu Planner
	guest	Guest List
Trip planner	event	Timeline
	time	

Table 6. Some new, student-designed representations based on observed referent types.

that more closely matched the access patterns of the information they encoded. For example, the "wedding dinner" group examined closely the conversations their users were having while planning the (theoretical) dinner. They found exchanges about seating to be a frequent occurrence, with many referents of type "guest" all connected together in a discussion about who to seat where at a particular table. As a result of this they designed a tailored Seating Chart representation, which vastly simplifies negotiation about seating by providing a persistent, shared representation for table and guest referents.

The redesigns were much better than those produced by earlier classes. The success stories from the Fall 2003 HCI class stand in contrast to results from prior sessions of the same class. In both the Spring of 1999 and the Fall of 2000, we taught a similar class and asked students to complete a very similar assignment; students were asked to design and implement a basic groupware system, demonstrate it to users, and redesign it based on user feedback. Although students were exposed to a variety of design techniques, including a thorough treatment of distributed cognition, and coverage of Shneiderman's prescriptions for software design [25], one of the things they struggled with was how to convert these general observations into specific design changes. In contrast, the methods presented in this paper aided students in investigating and redesigning their domains because of the structure they provided. By telling these novice analysts what to look at and what to look for, and then what to do with it, the methods helped guide analysts through the design process.

Reproducing results

To test their ability to reproducibly apply the methods, the students of the Fall 2003 class were asked to perform a referential structure analysis of four standard transcripts of GHT data. Parts of the GHT study had been discussed in class on several occasions, so although they had not seen the specific data they were given, the students were familiar with the domain. After the analyses were performed, we engaged the class in a discussion of the results and methods from this analysis, which yielded strong positive feedback about the utility of the method. In addition to providing students with unambiguous feedback about their ability to perform the analysis correctly, this exercise allowed us to test the inter-coder reliability of the methods.

Each transcript was analyzed by five pairs of students. The resulting analyses were qualitatively similar, though there were minor variations in results from group to group. To quantify the agreement, we used Cohen's Kappa, a standard method for comparing two or more analyses of a single set of data [4]. While it is meant to be applied to a situation where independent analysts are sorting items into one of a number of categories, with some adaptation we were able to apply it to our data, even though the task was not a strict category-assignment task. Kappa values for each group ranged from 48% to 71% — fair to moderate agreement — with an overall average of 62%, moderate agreement. This is a good result for informal data of this nature, and indicates that the methods can be used to achieve reproducible results.

CONCLUSIONS

In this paper we have presented a method for extracting observations from ethnographic study of a work domain and using them to help design a new system of representations to support that work. This approach examines the references made by participants to reveal the way different types information is passed around and stored in representations. By matching the ways that users handle different types of information, an analyst can recommend specific representations to mediate that information, based on the observed properties of the information and the features of the representation. We have shown that this approach can be used to design effective groupware for a system. Experiments have also demonstrated that the insights these methods generate into the justifications for use of representations are strong enough to be used both to help design groupware which supports collaboration, and to predict mismatches between representation design and task requirements.

We have also shown that the methods are teachable and reproducible: students were able to successfully apply the methods in the course of analyzing their own domains. The methodology provided the students with a step-by-step procedure to use to refine their applications, giving them guidance as to what portions of the interaction to address. When applied to standard transcripts, students were able to come up with comparable results. Finally, the students demonstrated that the methods can be applied to and generate redesign recommendations for a wide variety of domains.

ACKNOWLEDGMENTS

Support. This research was supported by the Office of Naval Research under grants No. N00014-96-1-0440 and N66001-00-1-8965. Additional support came from NSF grant EIA-0082393.

Thanks goes to Josh Introne, Seth Landsman, and others for comments and continued collaboration in support of this paper. Special thanks also goes to the Fall 2003 class of CoSci 111a, and all our experimental subjects, for providing data and extensive feedback on these methods.

REFERENCES

1. Alterman, R., Feinman, A., Introne, J., and Landsman, S. Coordinating representations in computer-mediated joint activities. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society* (Hillsdale NJ, 2001), Lawrence Erlbaum Associates.

2. Clark, H. *Using Language*. Cambridge University Press, New York, 1996.

3. Clark, H., and Brennan, S. Grounding in communication. In *Perspectives on Socially Shared Cognition*, J. Levine, L. Resnik, and S. Teasley, Eds. American Psychological Association, New York, 1991, pp. 127–149.

4. Cohen, J. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* (1960), 37–46.

5. Ellis, C., Gibbs, S., and Rein, G. Groupware: Some issues and experiences. *Communications of the ACM 34* (1991), 38–58.

6. Feinman, A., and Alterman, R. Discourse analysis techniques for modeling group interaction. In *Proceedings of the Ninth International Conference on User Modeling* (New York, 2003), Springer-Verlag, pp. 228–237.

7. Foster, G., and Stefik, M. Cognoter: Theory and practice of a collaborative tool. In *Proceedings of the 1986 ACM Conference on Computer-Supported Cooperative Work* (Austin TX, 1986), ACM Press, pp. 7–15.

8. Goodman, B., Linton, F., Gaimari, R., Hitzeman, J., Ross, H., and Zarrella, G. Using dialogue features to predict trouble during collaborative learning. *User Modeling and User-Adapted Interaction 15*, 1 (March 2005), 85–134.

9. Grosz, B., Joshi, A., and Weinstein, S. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics 2*, 21 (1995), 203–225.

10. Grosz, B., and Sidner, C. Attention, intentions, and the structure of discourse. *Computational Linguistics 12*, 3 (1986).

11. Hirst, G. *Anaphora in Natural Language Understanding*. Springer-Verlag, 1981.

12. Hollan, J., Hutchins, E., and Kirsh, D. Distributed cognition: Toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction 7*, 2 (2000), 174–193.

13. Hutchins, E. *Cognition in the Wild*. MIT Press, Cambridge, 1995.

14. Hutchins, E. How a cockpit remembers its speeds. *Cognitive Science 19* (1995), 265–288.

15. Kraut, R., Fussell, S., and Siegel, J. Visual information as a conversational resource in collaborative physical tasks. *Human-Computer Interaction 18* (2003), 13.49.

16. Langton, J., Hickey, T., and Alterman, R. Integrating tools and resources: a case study in building educational groupware for collaborative programming. *The Journal of Computing Sciences in Colleges 19*, 5 (2004), 140–153.

17. Lockman, A., and Klappholz, A. Toward a procedural model of contextual reference solution. *Discourse Processes 3* (1978), 25–71.

18. Martin, D., and Sommerville, I. Patterns of cooperative interaction: Linking ethnomethodology and design. *ACM Transactions on Computer-Human Interaction 11*, 2 (2004), 59–89.

19. Olson, J., Olson, G., Storrosten, M., and Carter, M. How a group-editor changes the character of a design meeting as well as its outcome. In *Proceedings of ACM CSCW'92* (1992), pp. 91–98.

20. Rogers, Y. Coordinating computer-mediated work. *Computer Supported Cooperative Work (CSCW) 1* (1993), 295–315.

21. Rogers, Y., and Ellis, J. Distributed cognition: an alternative framework for analysing and explaining collaborative working. *Journal of Information Technology 9*, 2 (1994), 119–128.

22. Sacks, H. *Lectures on Conversation*. Basil Blackwell, Oxford, 1992.

23. Sacks, H., Schegloff, E., and Jefferson, G. A simplest systematics for the organization of turn-taking for conversation. *Language 50* (1974), 696–735.

24. Schegloff, E. Conversation analysis and socially shared cognition. In *Perspectives on Socially Shared Cognition*, J. Levine, L. Resnick, and S. Teasley, Eds. American Psychological Association, New York, 1991, pp. 150–171.

25. Shneiderman, B. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 3rd ed. Addison-Wesley Publishing, Reading MA, 1998.

26. Suchman, L., and Trigg, R. Understanding practice: Video as a medium for reflection and design. In *Design at Work*, J. Greenbaum and M. Kyng, Eds. Erlbaum, Hillsdale NJ, 1991, pp. 65–90.

27. van Deemter, K., and Kibble, R. What is coreference and what should coreference annotation be? In *Proceedings of the ACL'99 Workshop on Coreference and its Applications* (1999), pp. 90–96.