HUMAN-COMPUTER INTERACTION, 1994, Volume 9, pp. 427–472

Characterizing the Sequential Structure of Interactive Behaviors Through Statistical and Grammatical Techniques

Gary M. Olson, James D. Herbsleb, and **Henry H. Rueter**
University of Michigan

ABSTRACT

Statistical and grammatical techniques are reviewed as an integrated approach to exploratory sequential data analysis (ESDA) for categorical data. The first step is the identification and validation of the categories to be analyzed. The main statistical techniques discussed are log-linear modeling and lag sequential analysis. These methods allow for the statistical evaluation of a wide range of general and specific hypotheses about sequential structure. Grammatical techniques based on

Gary M. Olson is a psychologist interested in the contributions of cognitive, social, and organizational research to the design and evaluation of computer systems; he is Professor of Psychology and Director of the Collaboratory for Research on Electronic Work at the University of Michigan. James D. Herbsleb is a psychologist and computer scientist interested in the effects of adopting software-engineering process and technology innovations; he is a Member of the Technical Staff in the Empirical Methods Project of the Software Engineering Institute. Henry H. Rueter is a mathematical psychologist with an interest in syntactic pattern recognition; he is a Research Scientist with the CREW group at the University of Michigan.

CONTENTS

definite-clause grammars are described and illustrated, and the complex issue of measuring the goodness of fit of a set of patterns is discussed. Throughout the article, examples from our own research illustrate how the various techniques are used, especially in concert, while carrying out ESDA. In Section 6, several other human–computer interaction and computer-supported cooperative work applications of these techniques are discussed.

1. INTRODUCTION

Human behavior exhibits sequential organization in a wide range of circumstances. Capturing this sequential structure can be a useful step toward understanding how people accommodate to different situations. In our case, we are interested in how people use computing and communications technology in various work settings. Often researchers assess the effects of such technology on work by focusing on the quality of the work done, the satisfaction of those using the technology, or the long-term organizational acceptance of the technology. Much less often do investigators focus on the details of the process of interacting with the technology or on how the organization of the work itself is affected. Interactive technologies are likely to influence the sequential organization of behavior, and this is likely to be a major source of clues as to why the technology affects work the way it does.

The sequential structure of interactive behavior can be characterized in many ways, both qualitative and quantitative, and each of the many methods has its strengths and weaknesses (see van Hooff, 1982, for a good overview of a variety of sequential analysis techniques). In this article, we focus on two classes of methods that we have found to be especially useful–*statistical techniques* that analyze the structure of behaviors and *grammatical techniques* that use rewrite rules to describe structure in sequences. We describe several techniques in each of these classes, illustrating their use with material from our program of research on computer-supported cooperative work (CSCW). Throughout, we stress that these two classes of techniques work well together. In the final section of the article, we describe the broader usefulness of these techniques for empirical studies of human–computer interaction (HCI).

The analysis of sequential behavior begins with the collection of a stream of relevant behavior. There are two broad classes of behavior that get used extensively in HCI and CSCW. First, we can record interactive behaviors with a rich medium, such as videotapes or audiotapes. For example, we might have an individual subject think aloud while interacting with technology, or we might transcribe the conversation that occurs between individuals in a group during normal social interaction. In order to use the techniques described in this article, such raw behavioral data have to be represented as a sequence of codes that capture some aspect of the behavior. We might transcribe the speech and classify utterances into categories by their content or form. We might code such nonverbal behaviors as gestures, gaze direction, or posture. All this kind of coding is labor intensive. A second kind of sequential data is a log of behaviors automatically captured by software as subjects use the software. In interactive systems, this includes considerable information about what a user or users did. These data do not usually require the initial heavy investment in coding as does the first type, because much coding can be done by computer algorithms or heuristic recoding.

We have used both forms of data in our work. Our basic strategy is to videotape sessions of small groups working on collaborative problem-solving. We have carried out studies of work both in the field (G. M. Olson, J. S. Olson, Carter, & Storrøsten, 1992) and in the laboratory (J. S. Olson, G. M. Olson, Storrøsten, & Carter, 1992, 1993), mostly looking at the process of software design.

In this article, we present the kinds of methods we have used for exploratory sequential data analysis (ESDA; Sanderson & Fisher, 1994), illustrating them with data from our field and laboratory studies of design. Our discussion has four steps: how the core categories get identified; statistical approaches; grammatical approaches; and using multiple methods to blend theory and inductive analysis in order to achieve understanding of the target behavior. Along the way, we offer methodological advice on how to carry out each of these steps. Toward the end of the article, we describe a series of further examples of how these techniques could be used in HCI and CSCW research.

2. THE STUDIES

Before turning to the details of our methods for ESDA, we provide more detail about the data we use to illustrate our approach. We use data drawn from two design studies we carried out. The details of methods and results were reported elsewhere, but here we provide enough information about each so that the reader can understand the details of our approach to ESDA.

2.1. The Field Study

We collected a large sample of videotapes of software-design meetings at two companies, company A and company B (G. M. Olson, J. S. Olson, Carter, & Storrøsten, 1992). A sample of 10 meetings was drawn from four different projects in the two companies. Each meeting lasted 1 to 2 hr, each was part of software-design projects that lasted for many months, and each included a mixture of requirements analysis and high-level design. Each meeting had three to seven participants, mostly experienced designers. The major focus of each meeting was software design, but, because these meetings were parts of ongoing projects, there were also discussions of project management–that is, how the current discussion in this meeting fit into the overall project. The only elements of technology used in these meetings were standard meeting-room tools such as whiteboards and flipcharts. The participants often took notes and also shared lists, sketches, and drawings that they had prepared before the meetings.

Each meeting was transcribed, and we used the transcripts to carry out a variety of analyses of design behavior, as we describe later. Our ESDA analyses consisted of coding conversational segments and analyzing the sequence of these activity coding categories.

2.2. The Laboratory Study

The second example we use is data from a laboratory study of design (J. S. Olson et al., 1993) in which groups of three subjects who knew one another and who had previously worked together as a group carried out a design task that was intended to mimic key aspects of design that we had seen in our field study. They were asked to come up with the initial requirements for an automatic post office (APO)–a collection of postal services offered through a stand-alone device similar to an automatic teller machine (ATM). It was to be designed so that a working prototype could be built by a company of 30 people within a year. Each group worked on this problem for 90 min. Before this task, the groups had worked on two 20-min tasks to get familiar with the laboratory setting.

The groups met in a special meeting room, the Collaboration Technology Suite (G. M. Olson, J. S. Olson, Killey, et al., 1992). Networked Apple Macintosh computers with dual-page displays were embedded in the meeting-room table. There were also large whiteboards and other conventional meeting technologies available. Half the groups ($n = 19$) shared their work using the whiteboard and paper and pencil; the other half ($n = 19$) used a group editor, ShrEdit (McGuffin & G. M. Olson, 1992). ShrEdit allowed subjects to view the same documents on their screens and to edit these documents freely at any time. Each subject could view any portion of the document they liked, although ShrEdit did provide some simple mechanisms for coordinating their views if subjects wanted to be sure to look at the same things at the same time.

We videotaped all 38 sessions, transcribed them, and coded them using the same scheme we had used earlier in our fieldwork. This scheme had to be modified slightly for the laboratory study because some new behaviors were possible here that were not seen in the fieldwork.

3. IDENTIFYING THE CATEGORIES

We focus our discussion of sequential structure on categorical data. We assume that the data to be analyzed consist of a string of activity codes, or a series of categories, that represent some kind of interesting classification of the behaviors under investigation. We expect that new categories might get created as part of the sequential analysis through processes like rewriting–analogous to the introduction of nonterminal symbols in linguistic analysis (Smith, Rooks, & Ferguson, 1989). In other words, the categories themselves may have structure to them. But, we explicitly ignore other kinds of data–such as the quantitative data that might be used to carry out a time-series analysis (Chatfield, 1989; Losada, Sanchez, & Noble, 1990).

When analyzing categorical data, the first step is to create the categories (see discussions by Bakeman & Gottman, 1986; Ericsson & Simon, 1984). Two major factors ought to govern the creation of categories–theoretical interest and objectivity. Theoretical interest depends on the domain. Moreover, for any type of interactive behavior, many possible categories exist at any of several levels of granularity. Objectivity is the traditional concern with the validity and reliability of the categories. How easy these are to achieve depends again on the domain and the complexity of the phenomena. However, sequential analysis is pointless if categories do not represent the thing they claim to represent or if two observers cannot agree in their coding. Not only must reliability and validity be assessed, they must also be reported, so that readers know the empirical status of the categories used.

In some cases, the initial categories may be quite straightforward. For instance, Lansman, Smith, and Weber (1990) used events that had been recorded by the software in a log of activity during an interactive session. This included things such as commands issued, selections made, and characters typed. Issues of reliability and validity were quite straightforward. More difficult are those cases in which the codes are based on an interpretation of the original behavior captured in a rich medium. We briefly illustrate this process with an example from our own research (see also Bakeman & Gottman, 1986).

Our goal in developing this coding scheme was to characterize the content of design discussions in the field as reflected in transcripts of our videotapes of design meetings. We were interested in the interplay of design problem-solving and social management of group activity. We drew upon two literatures for initial ideas for categories–design rationale (Moran & Carroll, in press) and group-management processes (Poole & Hirokawa, 1986; Putnam, 1981). These gave us an initial set of categories that made theoretical sense to us. However, the fit of these categories to our data was imperfect. Some behavior fit no category, and other behavior was difficult to classify reliably. This led to the creation of a new set of categories through combining, splitting, adding, and removing the original categories. For a category to survive in our coding scheme, it had to satisfy the joint criteria of theoretical interest and objectivity. The categories had to be precisely defined, with specific operational guidance to coders about how to apply them. The categories had to be tried on meeting transcripts from a variety of different sources so that we would be sure our scheme was not linked to the specific characteristics of one meeting or one set of participants. This iterative process continued until the categories achieved stability. The upper portion of Figure 1 shows the initial set of stable categories.

At this point, we coded the sample of 10 design meetings from the field. This led to a major revision of our initial coding scheme. We found that one category of behavior, clarification, occurred 33% of the time in our sample of meetings. In an effort to more fully understand its role, we took what had originally been a generic clarification category and revised our coding scheme so most clarification activity could be associated with its target activity.

The lower portion of Figure 1 presents the 22 categories that resulted from our modifications. Complete descriptions of the categories and their

Figure 1. **Categories of design-meeting activity.**

Initial Categories	
Issues	Digression
Alternatives	Goal
Criteria	Walkthrough
Project management	Clarification
Meeting management	Other
Summary	
Revised Categories	
Issues	Digression
Clarification of issues	Clarification of digression
Alternatives	Goal
Clarification of alternatives	Clarification of goal
Criterion	Walkthrough
Clarification of criterion	Clarification of walkthrough
Project management	Other
Clarification of project management	Clarification of other
Meeting management	General clarification
Clarification of meeting management	Artifact clarification
Summary	Clarification of summary

reliability are in G. M. Olson, J. S. Olson, Carter, and Storrøsten (1992). We achieved high degrees of reliability in our coding: For the revised categories, we achieved interobserver correlations for the times in and transitions among these categories ranging from .83 to .99.

For the laboratory groups, we added several additional categories. Because the groups had to produce an output by the end of their sessions, we coded discussions of the organization and wording of the final document as the category of plan and write. Groups using ShrEdit occasionally were confused by the technology or talked about strategies for how to use it. For this, we used the categories of technology confusion and technology management, respectively. This meant that the coding scheme for the laboratory groups had 25 categories. We achieved sufficient levels of reliability for these categories as well.

The categories we settled on represent a particular level of analysis that we felt was interesting. There are many other levels we could have chosen, and, indeed, we examine some others in later analyses. The present categories are a fairly high-level classification of the content of utterances. They do not reflect which person in a meeting made the contribution, how much time the contribution consumed, or what specific topic was discussed. There are theoretical reasons why all these might be interesting to analyze. Later, when we describe some of the details of our grammatical analyses of our materials, we present excerpts from our transcripts that show how coding was performed (see Figure 11).

4. STATISTICAL APPROACHES

We describe two broad classes of statistical techniques that we have found to be useful for the study of interactive behavior and that can be applied to categorical data–*log-linear modeling* (*LLM*) and *lag sequential analysis* (*LSA*).[1] LLM is a general technique for constructing and evaluating contingency-table data. It is especially useful for obtaining a global assessment of (a) the degree to which the data are sequentially structured rather than random and (b) how that structure changes over time and across conditions. LSA is useful both as an exploratory and confirmatory technique for finding which particular events follow other events at frequencies greater (or lower) than chance. LSA has helped us develop grammars that capture sequential structure.

4.1. Log-Linear Modeling

LLM is a very powerful, general technique for the analysis of contingency-table data (see, e.g., Fienberg, 1980). A familiar example is the test of stochastic independence of two variables by calculating chi square for a contingency table. LLM involves first creating a model by generating expected values for a contingency table and then testing the fit of observed values to this model. LLM is applied to sequential analysis (see Gottman & Roy, 1990) by constructing contingency tables[2] that represent the frequencies of sequences of categories and then by comparing the data to expected frequencies generated under some model. The approach resembles the general linear model, which forms the basis for analysis-of-variance techniques. Main effects and interactions of the researcher's choosing can be added in, and the expected values can be calculated accordingly.

Selecting an appropriate model depends, of course, on the research question one wants to ask. If one has a specific comparison in mind, then the model is constructed in order to test that comparison. In our work, as we discuss later in this section, we were interested in testing a three-way interaction. So, we generated expected values based on a model that included main effects for all three variables, as well as all two-way interac-

1. We focus here on methods we have found useful for our data and research questions. There are other approaches as well, such as those based on Markov modeling and information theory. See Gottman and Roy (1990) and van Hooff (1982) for overviews.

2. Some readers might argue that tables used to represent sequential data are not true contingency tables, because the observations are not strictly independent. However, this does not pose a major problem, because the G^2 statistic, discussed later in this section, is asymptotically distributed like a chi-square random variable (Anderson & Goodman, 1957; see Gottman & Roy, 1990, pp. 43–44).

tions. If the data do not fit the model, as they did not in our case, then the three-way interaction is statistically significant. Other researchers may wish to follow the general strategy of trying to find the simplest model–that is, the one with the fewest terms that provides a good fit. Given a table with a small number of dimensions, *N*, one may begin with a model that includes all terms except the *N*-way interaction.[3] If this model is a good fit, then terms can be removed until removing a term destroys the goodness of fit. The last model that fits well is probably a reasonably simple description of the data. There are several issues involved that this simplified description does not mention–that is, testing the statistical significance of the difference between two models and finding appropriate models when the number of dimensions is too large to make an exhaustive search practical (see Fienberg, 1980).

We have found LLM useful both as a way of investigating the relatively "global" sequential characteristics of the data and examining how these characteristics change across conditions. Because LLM is a very general and powerful analysis technique, one could also look at more complex effects (e.g., effects on sequential structure of interactions among different experimental effects or subject characteristics). It is also possible to partition the contingency table to focus on effects that are of particular interest (see, e.g., Fienberg, 1980, pp. 56–70).

One limitation of LLM is its sensitivity to low-frequency cells that can generate unacceptably low expected values. Unless tables can be collapsed or partitioned in appropriate ways, LLM is not appropriate for sparse matrices. In most sequential analysis applications of LLM, low expected values are generated by categories that simply are very infrequent in the data. In our own experience, these categories have not been particularly theoretically interesting, so, as we describe later, not much is lost by simply excluding those few transitions that involve these rare categories. But, of course, one should not exclude them purely on the basis of low frequency, because an event may be rare, yet crucial.

We present an example from our studies of how LLM can be used to understand the sequential structure of the data and to make interesting comparisons across conditions. We have data from the field and from two conditions in the laboratory, which we call *supported* (using ShrEdit) and *unsupported* (using only traditional tools).

We began by constructing a three-dimensional matrix of Antecedent Category × Consequent Category × Condition. The cell entries represent the observed frequency of a transition from a particular antecedent category to a particular consequent category in a given condition. So, for example, we observed 1,332 transitions from alternative to criterion in our

3. Testing the fully saturated model with all terms and interactions makes no sense, because a perfect fit is guaranteed. The expected values in the fully saturated model are always equal to the observed values.

unsupported laboratory groups, giving us one cell entry. The resulting matrix is quite large. Because there are 25 categories of activities and three conditions, the matrix is 25 × 25 × 3, or 1,875 cells. This matrix is the basis for all the analyses that follow, although we found it useful to collapse and partition it in various ways, as is to be described.

Search for Significant Structure

The goal of our first analysis was simply to determine if there was significant sequential structure in the data. To test this, we collapsed the data across the three conditions (initially, we were not interested in differences among them).[4] This gave us a 25 × 25 table of Antecedent Category × Consequent Category. If there is no sequential order, and the categories follow each other at random, then the values in each cell should reflect the joint probability of the antecedent and the consequent categories. This is the model that includes only main effects. If, on the other hand, there is a nonrandom sequential structure, the two variables will interact. So, we test for the existence of sequential structure by testing the fit of the main-effects model.

Problematically, however, there were many (244 of 625) expected values of less than 1, making the resulting statistical test suspect. Categories that have very low frequencies will produce very low marginal values for transitions to and from these categories. So, we elected to partition the table based on the overall frequencies of the categories. For further analysis, we selected the partition that consisted of the rows and columns where both antecedents and consequents were categories that had marginal frequencies greater than 100. We arrived at this cutoff value by balancing a desire to drastically reduce the number of cells with unacceptably low expected values with a desire to retain as much theoretically interesting data as possible.

By applying our cutoff value, we excluded 10 of the 25 original categories, which resulted in only 8 of 225 cells having expected values of less than 1. This seems like a drastic loss of data, because only 225 cells of the original 625 were retained. However, because only cells with low observed values were lost, 19,860 of the 20,608 original observations were retained in the remaining cells.

Testing this partition of the original table revealed a highly significant Antecedent Category × Consequent Category interaction, $G^2(196) = 8119.64$, $p < .001$. So, we concluded that, at least with respect to the transitions among the 15 highest frequency categories, there is definite evidence for sequential structure among adjacent categories.

4. Readers may recognize this as a test for the existence of a first-order Markov process, although we are deliberately ignoring potentially important properties like stationarity and homogeneity.

Detecting Sequential Similarities With Clustering

Next we wished to learn more about how the categories are related to one another in terms of their sequential structure. For example, we might find that two or more categories of activities are very similar both in terms of what triggers them and where they lead next, suggesting that they are substitutable. Or we might find that categories are very similar in terms of what triggers them but that they tend to lead to different places, suggesting they represent branch points in the process. Last, they might be triggered in different circumstances but have similar outcomes, suggesting they are a point at which different branches converge. The statistical conclusions might suggest unnoticed aspects of the process and encourage the investigators to return to the data to make more focused observations.

Hierarchical clustering (Johnson, 1967) is a reasonable technique for investigating similarities in sequential structure. A measure of similarity is needed first–which could be pairwise correlations calculated on lagged frequencies, on conditional probabilities or on z scores generated by a LSA. However, lagged frequencies and conditional probabilities are heavily influenced by the absolute frequencies of categories, so, even in randomly ordered data, there will be high dependencies between high-frequency categories and low dependencies between low-frequency categories. To avoid this, we used standardized residuals, which are the differences between the observed and expected values in a cell divided by the square root of the expected value. The expected values were generated by the main-effects model in the full 25×25 table already described. Standardized residuals represent deviations from chance–chance being what is predicted by relative frequencies of the antecedent and consequent of the transition. Two categories with similar sets of standardized residuals tend to deviate in similar ways from the pattern of transitions expected in a randomly ordered sequence.

Figures 2 and 3 show the results of two ways of performing hierarchical clustering on the standardized residuals of our data set, based again on the model that includes main effects for antecedent and consequent but that does not include the interaction term. Figure 2 is based on correlations between categories in the role of antecedent and so represents similarity based on what follows the category. Figure 3 is based on correlations between categories in the role of consequent and so represents similarity based on what precedes the category.

Interestingly, in Figure 2, there is a cluster near the center that includes all the clarification categories except clarification of walkthrough. So, almost regardless of what is being clarified, it appears that a similar pattern of subsequent activity will follow. This is quite interesting, because it seemed a priori just as likely that each clarification would cluster with its base category, so that emerging from clarification of goal would be more

Figure 2. Cluster on standardized residuals, aggregated data from field, supported, unsupported. Clar = clarification of, Mgt = management. Clustered based on similarity of what follows each category.

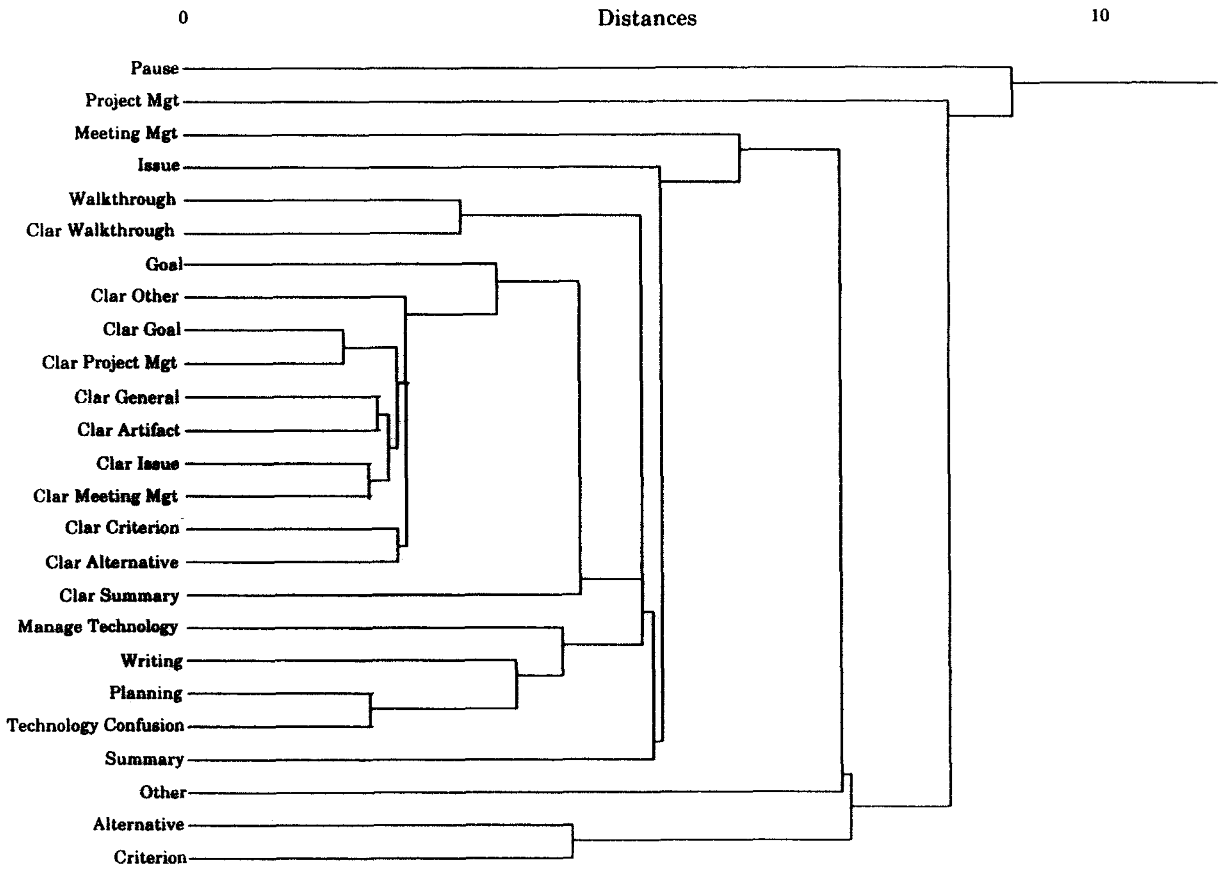
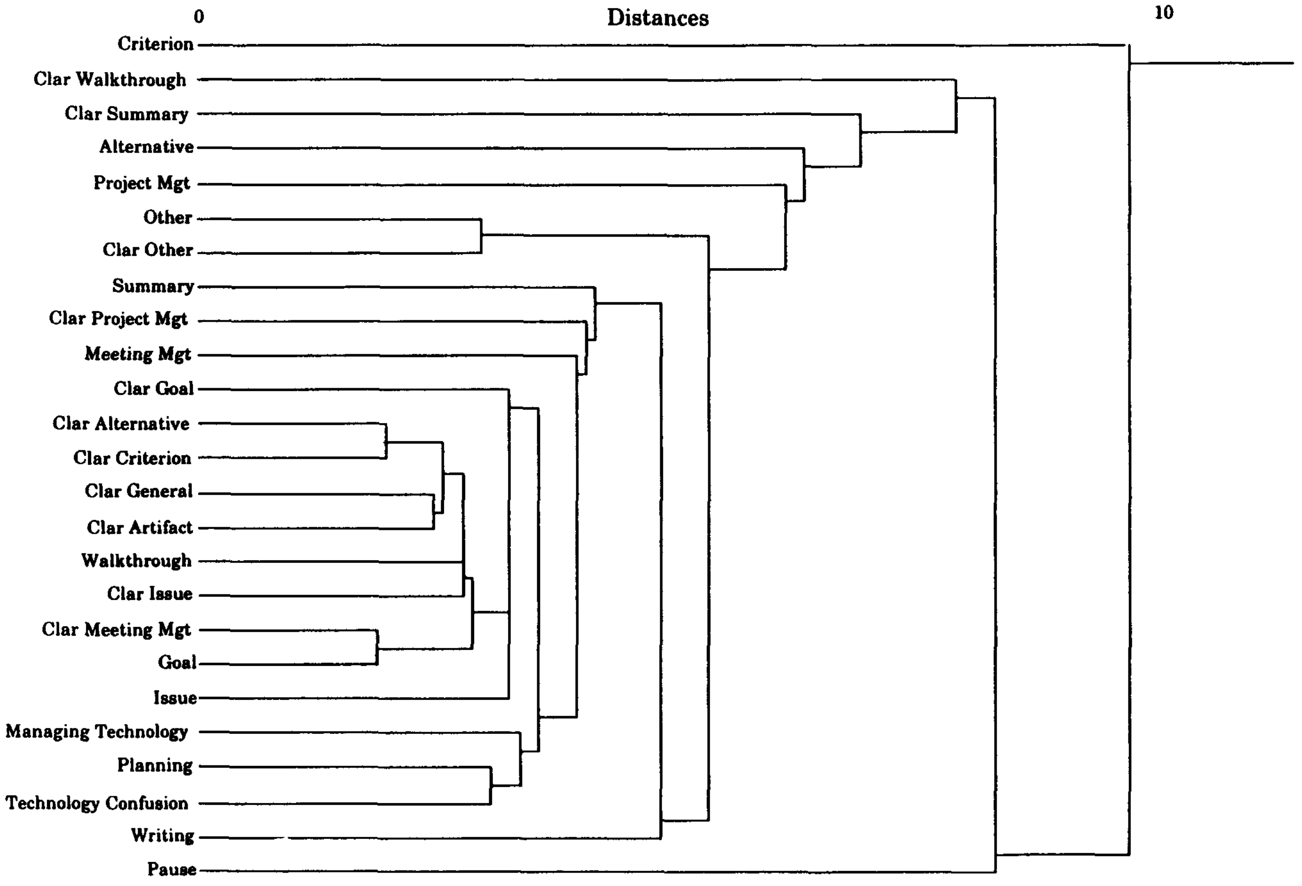


Figure 3. Cluster on standardized residuals, aggregated data from field, supported, unsupported. Clar = clarification of, Mgt = management. Clustered based on similarity of what precedes each category.



similar to emerging from goal than to clarification of project management. Overall, the picture suggests that the clarifications behave very similarly and are perhaps interchangeable in some sense. It has led us to begin to think of clarification, in general, as a distinct activity, rather than as a simple extension of the base activity. It also appears that many different branches of activity tend to converge when clarification is entered.

Figure 3 shows clusters based on similarities for each category in its role as consequent (i.e., in terms of what led to it). Here we find that the various clarifications are not tightly clustered and are in fact rather different from one another. The residuals reveal why: Almost every clarification tends to follow its base category (but often other categories as well), leading to dissimilar patterns of previous categories. There is an interesting exception to this, which is the high similarity between clarification of alternative and clarification of criterion. The residuals show that, unlike other clarifications, clarification of alternative tends to follow its base category much less often than would be expected by chance (this differs across conditions, as we point out later) and instead tends to follow criteria, generating a profile resembling clarification of criterion. This is another unexpected finding, suggesting that criteria help to clarify what an alternative is or implies.

These clustering analyses have several implications. For instance, one might want to rethink the categories and combine various ones because of their pattern of clustering. The clustering also provides clues to guide other analyses, such as our grammatical analyses.

Between-Condition Comparison: Field and Laboratory Data

We also used LLM to make between-condition comparisons. We intended that our laboratory task mimic the problem-solving aspects of design in the field, and LLM allows us to compare the sequential structure of activities in the two conditions. We were also interested in seeing if the sequential organization of the work in the laboratory was influenced by the kind of support provided. We had already found that the supported laboratory groups had better designs than the unsupported laboratory groups (see J. S. Olson et al., 1993, for details). Reasons might emerge in the sequential organization of their behavior.

In order to carry out this comparison of data from field, supported, and unsupported conditions, we return to the full three-dimensional contingency table already discussed, with 25 antecedent categories, 25 consequent categories, and 3 conditions as the three dimensions. The appropriate model includes all the terms except the one of interest. We included all three main effects, because we were not interested in differences attributable to simple differences in category frequencies or in differences generated simply by different numbers of transitions in the three conditions. All two-way interactions were also included, because we were not interested in simple differences in category frequency across conditions or in differences in antecedent frequencies across consequents. Because our hypothesis was that there are differences in sequential structure–or interactions of antecedent and consequent–across conditions, we were interested in the three-way interaction. So, the appropriate model is one that includes all main effects and two-way interactions.

But several problems stood in the way of testing this model immediately. First, the categories of digression and clarification of digression were simply removed from the data because they appeared to be both rare and uninteresting, and we counted the transition from whatever preceded to whatever followed each deleted item. The same approach was taken with the category of pause in most of our analyses because it often represented just a momentary lapse in the conversation, which we concluded was unlikely to have important causal properties.

As mentioned earlier, we introduced several categories to account for behavior seen in the laboratory but not in the field. These include planning the output for the laboratory task, writing the output, and so on. They could not be included in a comparison of laboratory and field, of course, but they nevertheless probably had a strong causal influence on what followed. Therefore, we excluded the preceding and following transitions by partitioning the contingency table and excluding that portion of the table that has one of these categories as a row or column variable.

We still had a $20 \times 20 \times 3$ or 1,200-cell matrix, however, and there were many expected values less than 1. So, we again excluded categories based on their frequency, this time requiring a raw frequency of at least 150 to be included.[5] This reduced the number of expected values less than 1 to only 13 of 192 cells. Although our partition rid us of nearly all marginal zeroes, 3 remained, which eliminated 6 cells from the analysis (see Fienberg, 1980, pp. 140–150). The three-way interaction was highly significant, $G^2(92) = 394.04$, $p < .001$, indicating that the sequential structure does change across conditions. Because the comparison of field and unsupported was of particular interest, we partitioned the table into a subtable consisting of these two conditions and a subtable consisting only of the supported condition and tested the three-way interaction for just the field and unsupported conditions. This was also highly significant, $G^2(45) = 271.83$, $p < .001$.

We investigated the nature of this difference by looking at identifying the categories that participate in the transitions that differentiate the two

5. We chose a higher cutoff here than in our preceding analysis because here we have a much larger table (1,200 cells in three dimensions, compared to 625 cells in two dimensions).

Figure 4. **Number of residuals with absolute values greater than 1.**

Category	Antecedent	Consequent
Alternatives	7	9
Criteria	4	1
Goal	2	4
Issues	11	8
Meeting management	8	7
Summary	8	6
Walkthrough	6	2
Clarification of alternatives	4	13

conditions. A large standardized residual indicates a substantial three-way interaction, which is our indicator of differences in sequential structure. So, for each category in its role as antecedent, we counted the number of standardized residuals (over all consequents) greater than 1. Then, for each category in the role of consequent, we counted the number of standardized residuals (over all antecedents) greater than 1. The tally in Figure 4 provides a rough indication of "where the action is" in the table–that is, categories in which both columns have many residuals with absolute values greater than 1. From this, we selected five categories for further analysis–that is, alternative, issue, meeting management, summary, and clarification of alternative. We partitioned the table and further analyzed the rows and columns where both antecedent and consequent came from this list, making a smaller table with 50 cells ($5 \times 5 \times 2$) of the original 128 ($8 \times 8 \times 2$). Testing the three-way interaction for this reduced table produced $G^2(16) = 198.49, p < .001$. So, even though this table holds fewer than half the cells of the larger table, it accounts for about three fourths of the G^2.

By again examining the standardized residuals, we identified particular transitions that are the major sources of difference. A reasonable rule of thumb is to look for rows in the table where the two conditions have standardized residuals with absolute value greater than 1 and are opposite in sign. This is a strong indication that the conditions are meaningfully different. Using this procedure, we find that, following an alternative, field meetings are much more likely to go to clarification of alternative, whereas unsupported meetings are more likely to go to almost anything else. In the field tasks, issues are more complex, and people have broader experience. In the simpler laboratory task, clarification of alternative is less often needed after an alternative. Interestingly, even though clarification of alternative is less likely to follow alternative in the unsupported condition, it is more likely to follow each of the other categories, suggesting that it can arise at any time in the unsupported meetings.

The residuals also show that, after an issue is discussed, in the field data the discussion is more likely to move on immediately to an alternative, whereas in the unsupported meetings it is more likely to move to anything else. This suggests a slightly different style of work.[6] We noted informally that people use issue statements to move a meeting along, as if saying, "Are we done with this one? Can we now do X?" The issues for which no alternatives are proposed are generally failed attempts to move along, and people go on with other activities instead of responding to the issue. There our sequential analysis pointed us back to the original data with some very focused questions about the role of issue in our data. The exploratory nature of ESDA is well served by these iterative cycles of learning from one's analyses and then returning to the data with more informed questions.

Between-Condition Comparison: Group Work With and Without Electronic Tools

Next we turned our attention to the differences between the two laboratory conditions. In order to undertake a more detailed examination of the sources of the differences, we added some of the data that were omitted earlier in order to make the three-way comparison of field, supported, and unsupported. Recall that several categories were omitted because they could occur only in the laboratory or potentially had more significance in the laboratory task. Because we wanted to compare the two laboratory conditions, we restored all the categories, and–by examining the expected values and eliminating low-frequency categories that generate them–we arrived at a selection of 10 of the original 25 categories. The final selection resulted in only 5 expected values of less than 1–out of 200 cells ($10 \times 10 \times 2$ conditions).

Recall that, in previous analyses, we found differences in output between the two groups, with the supported groups generating significantly better designs by focusing on fewer, but better, alternatives. We were now trying to account for this. We again generated for this new table a model that included all main effects and all two-way interactions in order to evaluate the significance of the three-way interaction. The result is highly significant, $G^2(81) = 186.05$, $p < .001$. As usual, we proceeded to a more detailed examination by inspecting the standardized residuals.

The differences are presented graphically in Figures 5 and 6. The arrows in Figure 5 represent transitions that tend to occur more often in the supported meetings, as measured by the standardized residual rule of

6. We are indebted to Judy Olson for the following interpretation.

Figure 5. Transitions that are more frequent in supported than in unsupported meetings, as determined by an examination of standardized residuals.

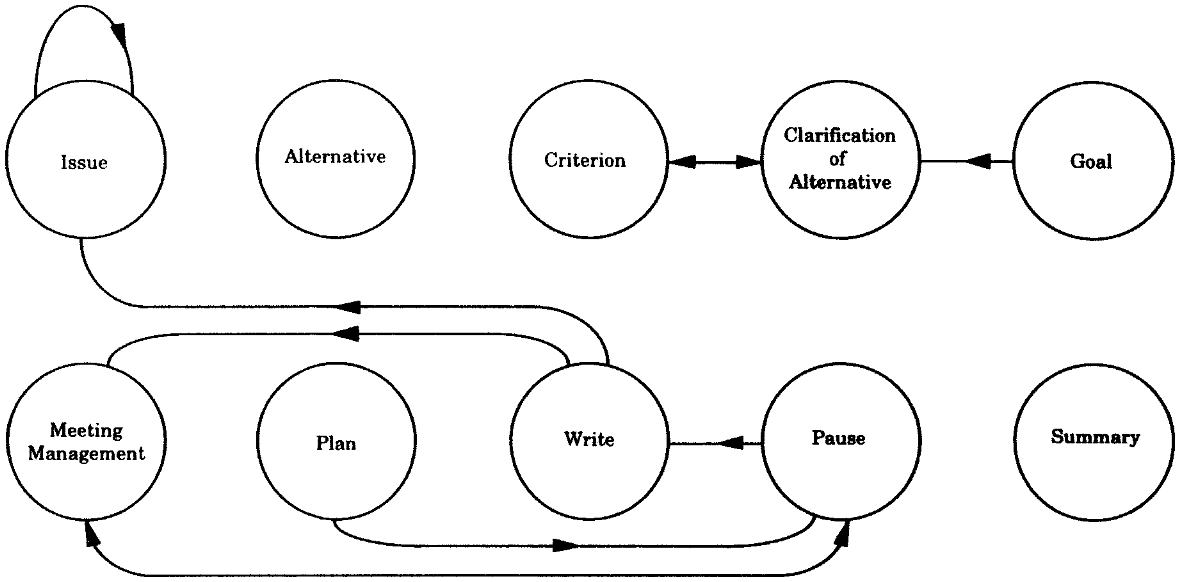
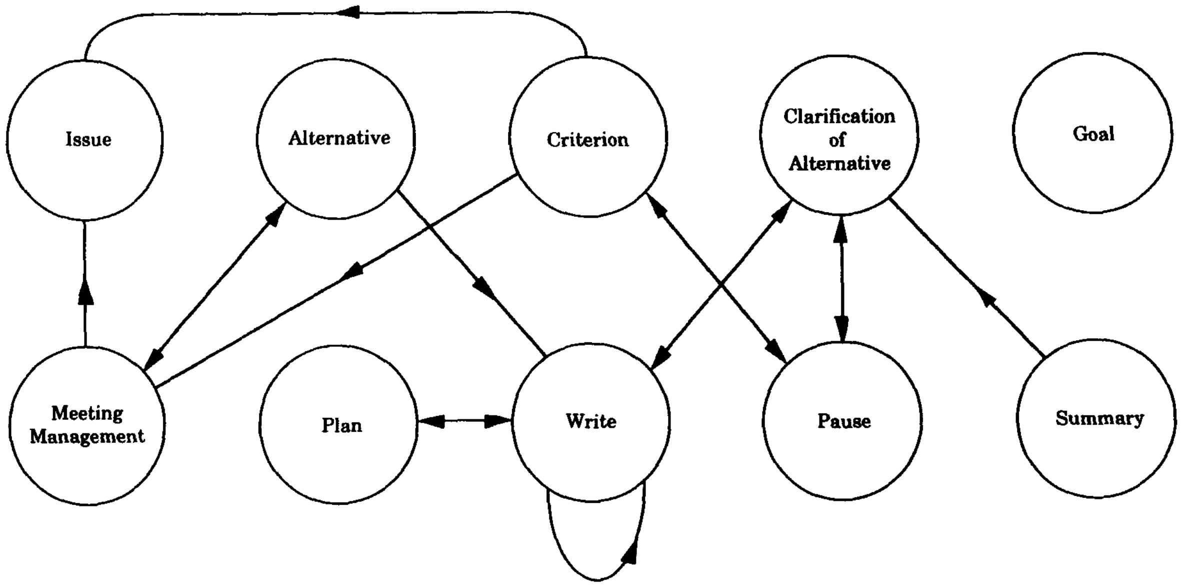


Figure 6. **Transitions that are more frequent in unsupported than in supported meetings, as determined by an examination of standardized residuals.**



thumb just mentioned. Figure 6 is a comparable diagram for transitions more frequent in the unsupported meetings. Notice in particular the vastly different role that clarification of alternative seems to play in the different conditions. In the supported group, clarification of alternative has a greater tendency to be triggered by discussions of criteria and goals and to lead more often to criteria discussions. In other words, the attempts to achieve a better understanding of the design options are more firmly embedded in discussions about what they are trying to accomplish (goal) and about how the quality of alternatives should be judged (criteria). In the unsupported group, on the other hand, clarification of alternative tends more strongly to be triggered by writing, by pause, and by summary and to lead to writing and pause. This suggests that alternatives are clarified more often in contexts associated with information capture–that is, trying to achieve a sufficient understanding to be able to summarize the alternative or render it in prose. The differences here suggest deeper processing in the supported meetings, which could play a major role in the quality of the design.

A clue as to the origin of this difference is supplied by our informal observations that the groups had very different styles of working. The supported groups tended to generate on-line notes throughout the entire task, and these notes evolved into the final document. The unsupported groups, on the other hand, made notes on paper or on the whiteboard but tended to compose the final document all at once at the end. It may be that the compose-as-you-go style encourages a relatively deep consideration of alternatives earlier in the process. This is consistent, of course, with the differences across conditions in the sequential structure surrounding clarification of alternatives. We are now preparing to compare the sequential structure of clarification of alternative that occurs early and late in meetings. So, our exploratory analysis again pointed us back toward the original data with very focused questions.

Practical Limits of LLM

We have found LLM to be a very powerful technique for determining that sequential order exists in a data set, for testing the significance of differences in sequential structure across conditions, and for locating particular transitions that contribute heavily to those differences. We have not found LLM to be as well adapted to the task of exploring longer sequences of categories in the data. We are interested, for example, in finding the sequential structure of core design activities–that is, how groups cycle through discussions of issues, alternatives, criteria, and clarifications. This involves looking at sequential relations of sequences longer than two units, which is better done with LSA and grammatical techniques, to be discussed.

4.2. Lag Sequential Analysis

LSA determines whether, given a particular category, subsequent[7] categories occur more or less often than one would expect by chance (Bakeman & Gottman, 1986; Gottman & Roy, 1990). The statistic with which we have the most experience is the z score proposed by Allison and Liker (1982):

$$z_1 = \frac{p_{B|A} - p_B}{\sqrt{\dfrac{p_B(1 - p_B)(1 - p_A)}{(n - k)p_A}}}$$

In this equation, $p_{B|A}$ is the observed proportion of occurrences of event B at lag k following event A. The quantities p_A and p_B are the observed proportions of events A and B; n is the number of events in the sequence. Other statistics can also be used to answer particular types of questions (see Wampold & Margolin, 1982).

In addition to investigating the statistical significance of immediately adjacent categories, LSA permits the calculation of z scores for states at arbitrary lags. In other words, one can calculate whether a given state is more likely than chance to follow another state after some number of intervening states. One can also look at patterns of relations among more than two states. For example, as an indirect way of testing the hypothesis that ABC is a frequently occurring pattern, one can calculate z scores for AB and BC (Lag 1) and A ... C (Lag 2). If all three are significant, one can claim that the data favor the hypothesis.

Calculating z scores for all combinations of states at one or more lags is also a useful exploratory technique for some purposes. Sequential connections observed in this way can serve as the basis for hypotheses to be explored in further studies. In addition, it is sometimes possible to identify "cliques" or clusters of states that tend to transition to one another, with relatively less frequent transitions to other states, perhaps even significant, negative z-scores between cliques. Such cliques may represent functional units in the process under observation. As a general caveat about post hoc analysis, if one sets alpha at .05 ($z > 1.96$), 1 of every 20 z-scores will be significant by chance. So, although this approach helps develop hypotheses, it is improper for testing them. One reasonable way of proceeding is

7. We and many other users of LSA are primarily interested in sequential relations between a category and what follows it. However, as a reviewer pointed out, these techniques can just as readily be used to examine what precedes a given category.

to use a sample of the data to develop hypotheses and then use the remainder of the data to test them.

Another caveat is that *z* scores do not necessarily reveal the sequences of states that are most frequent. Rather, they reveal states with sequential connections stronger (or weaker) than expected by chance. Consequently, one may neglect the most frequent pathways through the states that define a process. Researchers using LSA would be well advised to attend to the transitions that occur frequently, as well as to those with significant *z*-scores.

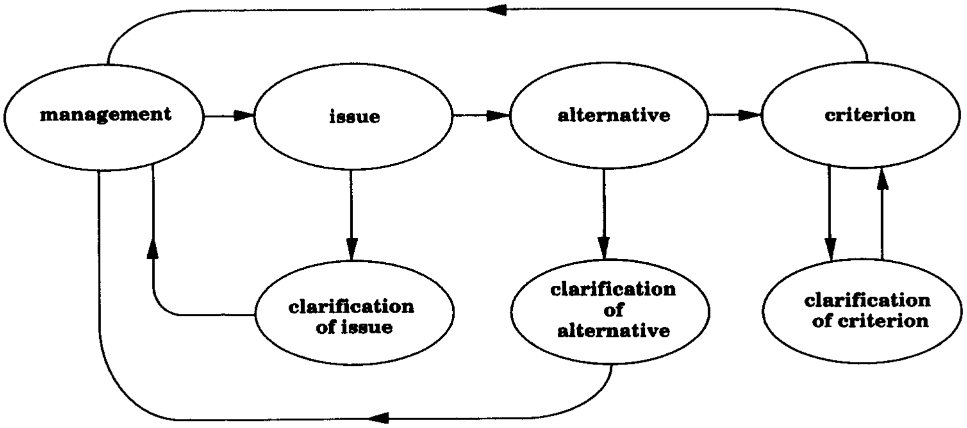
LSA can also be applied when there is more than one stream of data, as with a dyadic social interaction or a person interacting with a computer. LSA can be used to determine whether there is a cross-dependence of the behavior of one person at time $t+1$ on the behavior of the other at time t (Allison & Liker, 1982; Faraone & Dorfman, 1987). This may not be of particular interest for simple technologies, in which the user's behavior is closely linked to a dialogue specified by the software. But, in more open-ended situations, such as using a direct-manipulation interface, one might be able to study the effects of varying computer-system response delays by looking at the sequential organization of a user's behavior as it lags that of the computer.

LSA has played an especially critical role in helping us understand the longer sequential relations in our data. We focus here on the field meetings to illustrate our point. First, we examined how similar the meetings were to one another. For the set of 22 revised coding categories in Figure 1, the median correlation was .61 among the first-order transition matrices for the 10 meetings, suggesting that the meetings were quite similar. Thereafter, we used the data of all 10 meetings as though they represented the same process. We discuss other checks on this assumption as we go along.

An important discovery emerged through the use of LSA. We examined the statistically significant sequences in our data for lags up to 5, and we found that there were 25 significant transitions among the 6 design categories of issue, alternative, criterion, and their associated clarifications but only 2 from these categories to the remaining 16. Similarly, the remaining 16 categories had 79 significant transitions among themselves but only 11 with the 6 design categories. This pattern suggested that we had two functional clusters of activity (cliques) in our data–design (the 6 design categories) and management (the remaining 16 categories). We did not expect this. In particular, walkthrough appears to function in design meetings in a way very analogous to the other management activities, despite its surface appearance as a design activity. This illustrates how the functional significance of coding categories can be better understood by examination of their sequential structure.

Our clustering allowed us to simplify the categories for grammatical analysis. Because our principal focus was on design, we took the sequences

Figure 7. **The significant conditional transitions (Lag 1) among categories.**



of coding categories for the 10 meetings and replaced any sequence of the 16 management items into a single new category, management. This left us with just 7 categories–the 6 design categories plus our merged management category.

We repeated our statistical analyses with the reduced set of 7 categories. Figure 7 shows the significant conditional probabilities among the classes at Lag 1. These categories are arranged spatially to reflect the fact that management discussions seem to play an anchoring role in the design discussions, significantly often leading initially to an issue discussion. For example, all 10 meetings began with a management discussion immediately followed by an issue discussion. Overall, 54% of issue discussions were preceded by management (whereas only 12% were followed by management). So, the impression created by Figure 7–that issue discussions flow primarily from management discussions–appears to be correct.

Thus, purely on the basis of the use of statistical techniques, we were able to learn some surprising things about the sequential organization of our design meetings. But, we wanted to characterize the sequential structure in a way that would allow us to summarize in a few rules the kinds of sequential regularities that might exist in our data. For this purpose, we turned to rewriting techniques.

5. EXPLORING SEQUENTIAL STRUCTURE THROUGH REWRITING: USE OF GRAMMARS IN ESDA

A second important approach to the analysis of sequential structure that nicely complements the statistical techniques already described is syntactic pattern recognition. This approach uses grammatical formalisms to represent patterns (see Gonzalez & Thomason, 1978, for a good introduction to the field). Syntactic approaches to ESDA have been used in a variety of contexts for several years (see, e.g., van Hooff, 1982). As our discussion has pointed out, statistical descriptions of sequences are based on adjacency, on particular lags, or on influence that diminishes in a fixed way with distance. But, some kinds of sequences show embedded structures that cannot be captured by these statistical methods. Indeed, this observation was a cornerstone of Chomsky's (1957) approach to understanding linguistic structure.

Formal models or grammatical approaches have been used in recent years to define the behavior of user interfaces (see, e.g., de Haan, van der Veer, & van Vliet, 1991; Harrison & Thimbleby, 1990). One of the better known of these is Task-Action Grammar (TAG; Payne & Green, 1986), which defines user tasks in terms of values of semantic features and relates tasks to the actions that are the means to the ends of accomplishing the tasks. Grammars provide precise descriptions of selected aspects of the user interface (see, e.g., example TAGs for MacWrite, MacDraw, and others in Schiele & Green, 1990).

Our use of grammars is an integral part of an iterative approach to ESDA. In essence, we use syntactic rules to describe how structures of interest are built up using a series of rewrite rules–a common formalism in grammatical analysis. As we show later, this general strategy can be used in several ways. Theory, statistical analysis, and various other exploratory techniques suggest the syntactic rules. Issues of how well grammatical analyses account for sequential structure are less well worked out than in the case of statistical techniques, but we address these questions both abstractly and in relation to our empirical examples.

To describe patterns, we must have a pattern-description language. For this purpose, we found definite-clause grammars (DCGs) the most useful of the many available formalisms (Pereira & Shieber, 1987). Although many others such as regular expressions would also have served, we chose DCGs for several reasons. They are standard in virtually all dialects of Prolog, they can mix syntactic description with computation, they have the full expressive power of context-sensitive grammars, they have been widely used and extended in natural-language programming (NLP) work, and they are transparently easy to use. In addition, they have an easily penetrated literature aimed at all levels of expertise and at users from many backgrounds (e.g., NLP, chemistry, and constraint satisfaction). In short, they are very general and very powerful. We have used DCGs along with the programming languages Prolog and Lisp–a combination of formalism and implementation that has many good properties for ESDA.

We use DCGs to describe patterns by constructing a grammar for each pattern that associates a set of DCG clauses with a pattern name. In the following examples, we are assuming that the input to the rewriting process is a string of coding categories of the kind shown in Figure 1. For instance, the pattern *issue_alternative_criteria* describes a pattern in which an occurrence of the token *issue* anywhere in the data sequence is followed by the token *alternative,* which is followed by any number (at least 1) of *criterion* tokens. This pattern is represented recursively by the following six DCG clauses:

issue_alternative_criteria --> IAC^+ (1)
I --> *[issue]* (2)
A --> *[alternative]* (3)
C^+ --> CC^+ (4)
C^+ --> *C* (5)
C --> *[criterion]* (6)

These clauses express the fact that an *issue_alternative_criteria* pattern consists of an *I* pattern, followed by an *A* pattern, followed by a C^+ pattern (1). Proceeding recursively, an *I* is simply an occurrence of the token *issue* (2), an *A* occurs wherever an *alternative* token occurs (3), a C^+ pattern consists of either a *C* followed by another C^+ (4) or else a single *C* (5), and a *C* itself is an occurrence of the token *criterion* (6).

For our rewrites, we use three relatively simple types of rules. One is an *or* that just rewrites any occurrence of the disjunct as a single token. A second is a *plus* that rewrites any number of consecutive occurrences of a particular token as a single new symbol. The third is rules that *rewrite* sequences of literals as a single token. DCGs provide much more than this, of course, but these kinds of rules have been sufficient for our purposes. Indeed, the more complex the rules, the more difficult the results are to interpret.

We now consider how DCG pattern descriptions can be used to uncover patterns in data sequences. There are two basic approaches to this discovery process–parsing and rewriting. Most implementations of Prolog directly support parsing of token sequences using DCGs. A Prolog function is provided that takes as its input a sequence and a set of DCG clauses and yields as output a parse of the sequence, if one is possible. We used this built-in parsing facility to create another function that takes a sequence of DCG clauses, a pattern name, and a token sequence and replaces the longest subsequences that can be parsed according to the clauses with the name. We call this technique *rewriting* as opposed to *parsing* because its object is to rewrite pattern occurrences in the data sequence, not necessarily to provide a complete parse of it.[8]

8. Readers familiar with parsing techniques will recognize rewriting as a form of CYK or chart parsing; we say more about this connection later.

To illustrate this, an attempt to parse the sequence

issue alternative criterion criterion

using DCG clauses 1 to 6 will succeed, recognizing the (entire) sequence as an example of the *issue_alternative_criteria* pattern. We can, however, get basically the same result by successive rewriting according to the sequence of DCGs 6, 5, 4, 3, 2, 1–where we write *before --(N)-> after* to show the result of applying the DCG clause numbered N above:

issue alternative criterion criterion --(6)->
issue alternative C C --(4,5)->
issue alternative C^+ --(3)->
issue A C^+ --(2)->
I A C^+ --(1)->
issue_alternative_criteria

but the difference is that we have not committed ourselves to completely parsing the output. For instance, an attempt to parse the sequence

issue criterion criterion alternative

as an *issue_alternative_criteria* pattern would fail, but rewriting according to DCG sequence 6, 5, 4, 3, 2, 1 would produce some useful information–namely, that there is an occurrence of C^+ in the sequence:

issue criterion criterion alternative --(6)->
issue C C *alternative* --(4,5)->
issue C^+ *alternative* --(3)->
issue C^+ A --(2)->
I C^+ A

If we want to try to induce a complete grammar from data, we can use an iterative rewriting process, guided in various ways, to do this bottom-up. Tokens that remain unrewritten may be "noise," or they may be events with a significance that is not yet understood. By isolating these events, the process of rewriting helps to call attention to them.

Another approach is to seek completely automated techniques for inducing a grammar from a corpus of data (Miclet, 1986). This may not be as good an idea as it might seem, at least for ESDA. Some reasons for rejecting an automated approach are:

- The range of possible forms that grammars and patterns can take is too broad (see Grune & Jacobs, 1990, for examples). If the form of the grammar that is wanted can be quite narrowly defined, then automated induction might be worthwhile. This may not be possible, especially at the exploratory stage of data analysis.

- Sensitivity to noise in the data is a problem whenever one looks for patterns, and automated techniques tend to be especially vulnerable to noise. We overcome noisy data to some extent by putting the investigator directly into the loop, where domain knowledge can be brought to bear. (There is, however, the possibility of using stochastic grammars to address this problem; Fu, 1982, is a rich source for stochastic and other techniques for syntactic pattern analysis.)
- Perhaps the most important argument against automatic grammar or pattern induction was made in Pentland and Rueter (1992):

  > What makes a grammar useful? Like any model, it has to give you some kind of insight into the underlying process. It is not sufficient that a model does a wonderful job in fitting the data, it must also tell us something about the data. ...we have to look at these rewriting rules as [expressing] a *theory*: It has to make sense ... (p. 26)

  Furthermore, an investigator comes to understand the data through the process of investigation, not just through contemplation of the final result. This understanding is expressed in the rewrite rules–constituting, in effect, a set of rules analogous to the productions of a semantic grammar.[9]

If there are no criteria strongly limiting the form of the desired grammar, heuristic induction methods are essential for finding syntactic patterns in data sequences. Our own experience reinforces and strengthens this conclusion: We have not found general algorithms for the induction of grammars to be useful for ESDA.

In the remainder of this section, we describe the heuristic technique that has proved useful for us in ESDA. We begin with a set of seed patterns–which may be obtained from theory, statistical analyses, transition diagrams, simple eyeballing of the data, or a familiarity with the range of patterns expressible in one's chosen pattern language. So far, we have found ourselves using DCGs to encode simple, regular expressions, but DCGs (and other grammar formalisms, such as PATR) are also capable of expressing parameterized patterns as well as those with missing or shifted components. (See Abramson & Dahl, 1989, for an extended discussion of the uses of various logic grammars.) In addition, DCGs and other formalisms support the use of token "features" to pack more information into tokens (Gazdar & Mellish, 1989b; Shieber, 1986).

Finally, although we have confined ourselves to the use of context-free rewriting in our investigations so far, DCGs can also be used to express

9. We are indebted to an anonymous reviewer for pointing this out.

context-sensitive grammars. Context-sensitive grammars provide additional power by requiring that a rewriting rule be applied to a sequence of tokens only when that sequence is embedded in a specified context of surrounding tokens. In our own data, for example, we might expect an issue discussion that is part of a summary of a previous meeting to have a different character from an issue discussion on a current problem. The addition of more complex grammars remains a fertile area for further investigation.

We proceed by building up a list of patterns, expressed by DCGs, that we then iteratively rewrite in the data sequence of interest. First, the first pattern in the list is sought in the data sequence, and its first, longest extent is replaced by a token representing the name of the pattern. Then, the same pattern is applied to the rest of the sequence, and this is done iteratively, until the pattern occurs no more in the data sequence. Then, the next element in the pattern list is selected, and the data sequence is interactively rewritten in the same way. Eventually, the pattern list is exhausted, and the data sequence has been rewritten as far as it can be.

The order in which patterns are rewritten may affect the result: First, if a hierarchical pattern, like *issue_alternative_criteria* (as described earlier), is to be detected and rewritten in a data sequence, the constituents of the higher level pattern (I, A, and C^+ in this case) must have been previously found and rewritten. Clearly, by rewriting first *issue_alternative_criteria* and only then rewriting I, A, and C^+, one will never find any instances of the *issue_alternative_criteria* pattern. Second, prior rewriting of one pattern may preclude finding and writing another pattern. For instance, if we are looking for *issue_alternative_criteria* patterns, and some other pattern consumes *issues* (perhaps by rewriting *digression-issue* pairs as *digissues*), then those *issues* that are incorporated in *digissues* will not be available to form *issue_alternative_criteria* patterns.

Pattern discovery of the sort that we have described is a demanding task for a human being. It is therefore important to provide the investigator with as much interactive, computer-supported assistance as possible. Two techniques that we have found useful are:

- An automated view of the context of any set of tokens or developing pattern of interest. We have used a spreadsheet display as the interface to this function; the spreadsheet is linked to our Prolog and Lisp rewriting programs to allow the specification and viewing of the context around any set of tokens of interest, basic or rewritten. Through the use of multiple-column-sort functions in a spreadsheet, we can arrange the subsequences of data in various ways and use the pattern-detecting ability of the human eye along with the domain knowledge of the viewer to scan for useful and interesting rules for the next round of rewriting.

- Inspection of well-formed substring tables produced by the chart-parsing technique. In this technique, a parser is applied to the data, and a table is constructed that records the fate of all parsing attempts, both successful and unsuccessful. By examining the table, one can see where parsing attempts failed and why, as well as how successful patterns might be extended. A discussion of chart parsing and computer code can be found in Gazdar and Mellish (1989a, 1989b).

Although we prefer rewriting as an exploratory technique for sequential analysis for its efficiency, conceptual simplicity, and iterative nature, sometimes it may be helpful to provide a complete grammar for data sequences. In particular, for data generated by a relatively noise-free, highly structured process, parsing might be a natural and very useful approach. Possession of a full grammar lets one:

- Use the grammar to classify sequences by either recognizing or failing to recognize them. For meeting-state sequences, for instance, a grammar could be used to recognize "good" and "bad" meetings. Many of the techniques described in Fu (1982) can be brought to bear, such as measures of distance between a sequence and a given grammar and the use of error-correcting parsing techniques to deal with noise.
- Describe all the data, in a conceptually clean way, rather than piecemeal.

On the other hand, we might want to be content with less than a full grammar when:

- There is no discernible structure, and it would be useless to try to impose one. Two independently controlled sequences may be interleaved (e.g., in a CSCW application) and cannot be fitted by any rules without first being separated.
- We might not have enough information for a full grammar. Features or information might be missing from the categorized data.
- We might be interested in rewriting categories merely as data analysis categories, so we are not interested in a full parsing of the data but instead simply in the frequency of occurrence of certain patterns in the data, perhaps as a basis for comparison of different data groups.
- In exploratory analysis, emphasis on finding a complete grammar might lead to premature termination of exploration or to essentially empty patterns included only for formal completion of a grammatical form.

- Different aspects of a meeting might find expression in different kinds of grammars; in such a case, it would be inappropriate to search for a single overarching grammar.

The unearthing of patterns, especially with the noisy, relatively unstructured data we have dealt with, is very much an interactive and creative process. Finding patterns in sequences is plain, hard, creative work, despite the guidance provided by statistical analyses. The investigator is often confronted with an overabundance of information when undertaking the rewriting approach outlined here, especially at fine-grained levels of analysis. Dealing effectively with this amount of information requires interactive computer support, ranging from displays of pattern context within data sequences, to statistical summaries such as transition frequencies at various lags, to a chart-parsing display of why attempts to match patterns failed.

5.1. Measures of Fit and Significance

In assessing the success or usefulness of a syntactic analysis, the difference between the grammatical and rewriting approaches described in the previous section stand out clearly. When one fits a grammar to data, the sequence is either recognized by the grammar or not. On the other hand, when several rewritings are applied to a sequence, resulting in what we call a *translation,* we must judge the "goodness" of the translation. Are there other translations that are better for a particular data set? Would our translation have been just as effective when applied to random token sequences that keep the frequencies of tokens the same but eliminate whatever sequential dependencies might have been in the original?

The most fundamental concern is how much rewriting sequential data increases the investigator's understanding. Finding measures of relative "efficiency," parsimony, or similar criteria of the relative success of two different grammars in describing the same data set is difficult, because the wide range of grammars makes it difficult to see what a basis for meaningful comparison could be. Even when the comparison is limited to grammars of the same kind, problems remain that are similar to those in regression analysis: The more variables (rewriting rules) you use, the better you can do, but when should one stop adding variables (rewriting rules)? Have you used the best transformations (forms) for the variables (rewriting rules)?

Nonetheless, there are some rough figures of merit that might be useful. For instance, one could judge how much the translation has condensed the original data, because the degree to which we have rewritten the original sequence is a rough measure of how much we "understand" it. Of course, because even random data may show a certain degree of reduction, reduction in itself is not necessarily an indication of the presence of meaningful patterns. Interactive displays of relative condensation for real and random data can help the analyst judge the benefit of including a particular pattern in the rewriting scheme.

We can assess statistically the performance of a grammar (or set of rewriting rules) on a given data set by asking whether or not we achieve a "significant" reduction in the original data sequence. This will be the case if we have captured more regularities in our rewriting rules than there are in random data.

Suppose D is a data set–that is, a collection of sequences of tokens to be rewritten–and R is a sequence of DCG rules to be used to rewrite the tokens of D. Each sequence S in D will have a length before rewriting, *Before(S)*, and a length after rewriting, *After(S)*. Then, the percent reduction of S with respect to rewriting with respect to R is simply

$$\text{Reduction} = 100\left(\frac{Before(S) - After(S)}{Before(S)}\right)$$

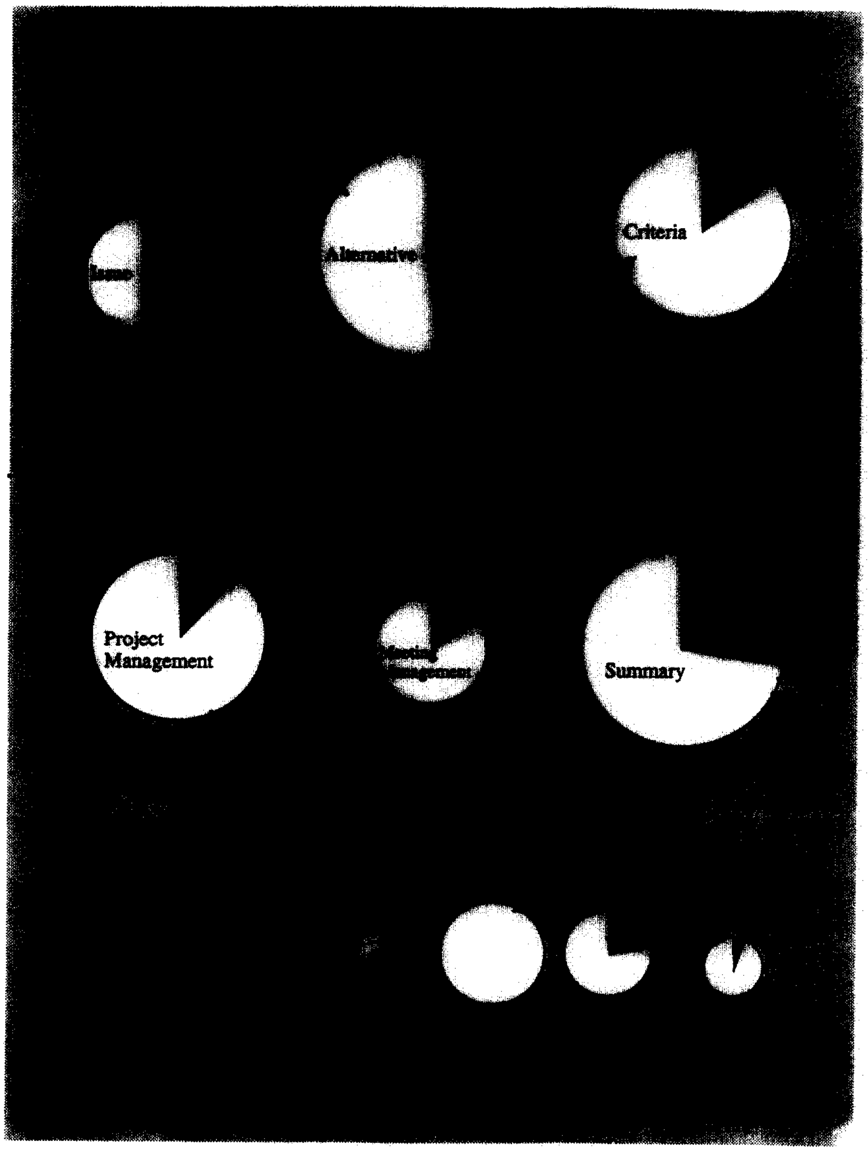
Let the average reduction for the set of strings D with respect to R be called *AverageReduction(D,R)*. This will be a number between 0 and 100. We wish to know whether we have achieved a statistically significant reduction of D through rewriting via R.

To determine the statistical significance of *AverageReduction(D,R)*, we can use a Monte Carlo estimation technique for the distribution of *AverageReduction(Population(D),R)*, where *Population(D)* is the set of all sets P, and $P = \{T{:}T \text{ is obtained from } D \text{ by permuting each } S \in D\}$. That is, the population P is the set of all sets of permutation of elements of D. The advantage of using permutations of actual data sequences is that we preserve the token frequencies and sequence length. We estimate the distribution of *Reduction(S)* by conducting an appropriate number of estimation trials, because P is virtually infinite for strings of more than a few tokens. An estimation trial consists of producing a random permutation of each element S of D to form *Random(D)* and then computing *AverageReduction(Random(D),R)*. The goal is to estimate the probability distribution function of the corresponding population parameter for the average reduction. With a sufficiently large sample (500 or so trials for .05 level for a one-tailed test of significance), we can estimate the cutoff points for various one-tailed tests of significance of any observed *AverageReduction(D,R)*. For example, if *AverageReduction(D,R)* is 78%, we can see whether less than 5% of the sample produced an average reduction of 78% or less.

5.2. Illustration of Grammatical Analysis

We have used the heuristics just described to analyze the organization structure of discussions in the collaborative design-meetings we have studied in the field. In particular, we have focused on the analysis of the

Figure 8. **Transitions among the 22 categories of design-meeting activity in field data.**



rewritten strings that consist of seven categories, the Lag-1 LSA results shown in Figure 8 for the field data. Our videos suggested that the meetings were quite informal and highly interactive, so we were skeptical whether there would be detectable structure. Our search for patterns was guided by three considerations:

- A priori conjectures as to how design as an activity might be organized–drawn from several sources (e.g., Moran & Carroll, in press).
- Statistical analyses of the dependencies among the coding categories, particularly LSAs that helped us identify potentially interesting clusters of categories.
- Various visual displays of the flow of activities among the categories (e.g., Figure 8).

Thus, using a combination of top-down and bottom-up heuristics, we looked for interesting pattern clusters in the data. Now we trace an example using our field data. When we found patterns of interest, we replaced them with a new category marker. For example, when we found that the design team had discussed one or more alternatives in a row, we replaced every occurrence of one or more *a* with some marker such as A^+. So, the sequence

mgt i a a c c a a a i a c c a

would become

$mgt\ i\ A^+\ c\ c\ A^+\ i\ A^+\ c\ c\ A^+$

As we proceeded, it was obvious that there were extended discussions of alternatives and criteria that were punctuated by discussions of issues and management. So, first we described the structure of these alternative and criteria discussions (see Figure 9). We searched for a variety of patterns in these four categories and ended up with a general pattern we called *Alt_Crit* that contained strings that shared the characteristic that discussions of alternatives tend to precede discussions of criteria.

Next, we developed patterns that used these *Alt_Crit* complexes as building blocks (see Figure 10). Issue-based design sequences consisted of an issue complex followed by an *Alt_Crit.* We called these *IAC episodes.* If these were preceded by a management discussion, we called these *MIAC episodes.* Last, we also defined patterns consisting of *Alt_Crit* complexes, without issues, that were preceded by management. We called these *MAC episodes.* An example of a sequence coded as MIAC is shown in Figure 11.

To evaluate the extent to which these patterns were truly structured, we compared the fit of our patterns in real meetings to their fit to 10 random "meetings" in which the base frequency of the 7 classes of activities matched that of the 10 real meetings. We then performed a similar rewriting exercise with each of these 10 random meetings and calculated comparable statistics about the fit.

Figure 12 shows the comparisons for the three patterns described in Figure 10. Overall, there are 96 IAC episodes in the real meetings but only 17 in the random meetings. The majority of the IAC episodes in the real

Figure 9. **Patterns involving alternatives and criteria *(Alt_Crit)*.**

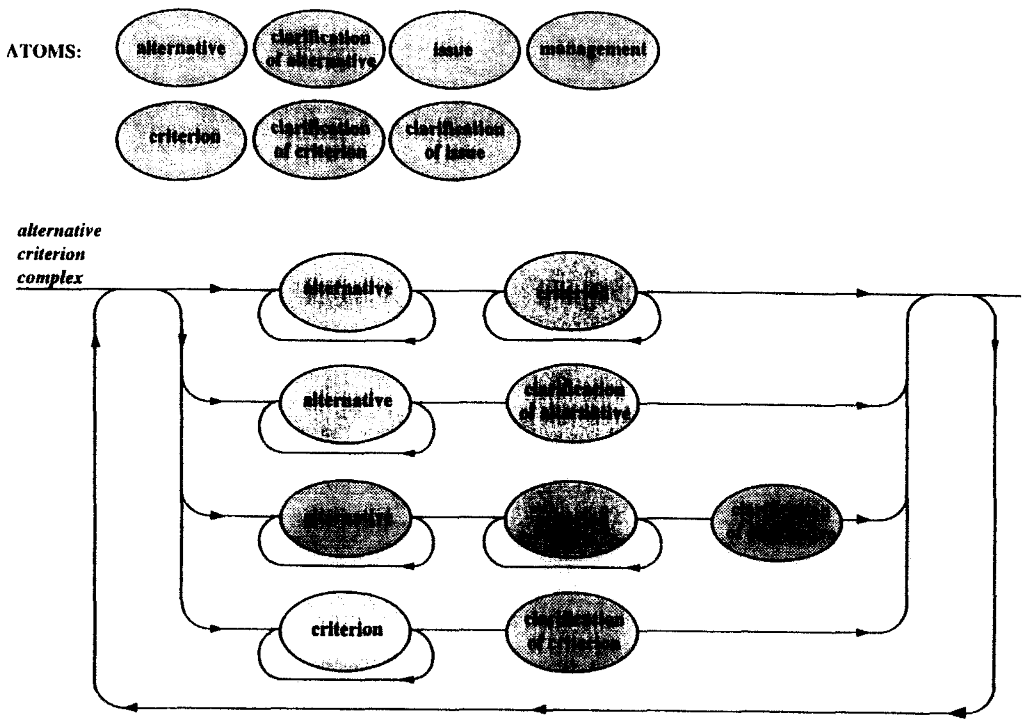
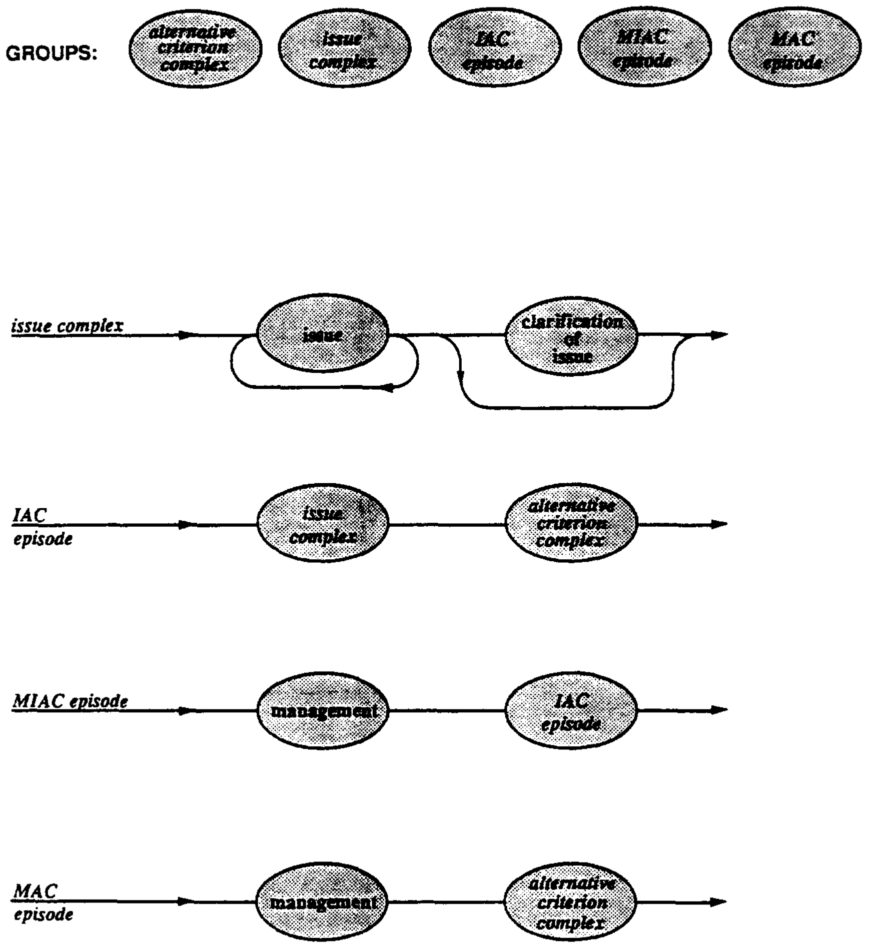

ATOMS:
alternative
clarification of alternative
issue
management
criterion
clarification of criterion
clarification of issue
alternative criterion complex
alternative
criterion
alternative
clarification of alternative
criterion
clarification of criterion

Figure 10. **Highest level patterns.**



data are contained in MIAC episodes, whereas less than one third of those in the random data are. Design sequences that do not contain an issue complex at their beginning are also more often preceded by management in the real data than the random (the MAC episodes), although this trend is not as strong as the MIAC episodes.

Good fit is also indicated by the fact that, in IAC episodes, 46% of the available appropriate tokens (the intermediate issue complex and *Alt_Crit* categories) were replaced for the real meetings, but only 11% of the same tokens were replaced for the random meetings; in MIAC episodes, 31% of available tokens were replaced for the real meetings, but only 1% were replaced for the random meetings. Thus, the real and random meetings

***Figure 11.* Example of an MIAC episode (nothing deleted).**

MIAC episode (management-issue-alternative-criterion-alternative-clar-alternative-alternative-criterion-[digression]-criterion)

Don:	I was just going to change the topic a little bit because I keep . . .	Meeting management --> mgt
	The thing you said after we started about how knowledge editing is if you do it alone, you haven't done . . . I mean, it's kind of fundamentally collaborative, right, because it's not just a process of building an artifact, it's a process of a group of people buying into a particular axiomatization of the world.	Issue
Bill:	Yeah, that's what's important, co-representation.	
Tom:	Yeah. One of the things I really liked about the Colab thing, I don't know if this is off the track, and what I always wanted to do when we were talking about putting this collaborations and this interaction thing is to have somebody else's frame up in the corner of my window so I could see what they were doing, kind of tiny, as I was doing whatever I was doing. That's what they did in the co-lab thing. In a sense, they had one little window up in the corner that was common to everybody and that you interacted with but you could still do your own interaction locally, and then maybe propagate it (unclear).	Alternative
Stu:	So that might be as easy as just like having a shared inspector or something, I mean, having one of the inspector designs that everybody could write into it.	Criterion
Bill:	It would be nice if you could have a window to another guy's interface to see what he's doing and somehow the thing would shrink in a cognitively appropriate way so that the meaningful, sort of the highlights would stand out and you could tell what he was doing, but a lot to the detail would fade.	Alternative
Don:	Say it again.	Clarification of alternative

(*Continued*)

Figure 11. ***(Continued)***

Bill:	It's the idea that you could say I want a window on Don's activity so it makes a little window of your whole screen but it's smart enough so that it doesn't just shrink pixel by pixel and then do the hypercard thing where you really can't see what your old frames are exactly but it somehow shrinks smartly so that maybe you just get the names of the units that are . . . and only slots that have been touched with the mouse and all other slots never get seen or something. That would be a way to sort of keep up with you but it would also be, I guess, according to your vision, I could actually go there and just do work. I could make things happen in your interface. If it's the co-lab idea, I have to be able to go up there, well, hot it's either a spy on your personal work area or it really is a group work area where I have license to go make modifications.	
Tom:	Yeah, that's what it seems like to me. It seems like you want the both of those things. You want a group area where everybody sees it and sort of everybody is interacting at the same time in that window. And then you want your own work area and then maybe you want to be able to see what somebody else is doing in their work area.	Alternative
	I don't know how you handle these on limited screens.	Criterion
Don:	A million pixels is not enough.	
Bill:	Well, you know though, we should be [digression] thinking . . . [I hate to think for the long term.]	(Deleted)
Don:	Golly, that would be bad.	
Bill:	But we will have the work surface someday, and I don't think we should really try and do it, but we should think with incompatible ways.	Criterion

Figure 12. **Comparison of pattern frequencies for real and random meetings.**

	Meetings	
Rewrite Rule	Real	Random
IAC episodes (not involved in MIAC episodes)	45	12
MIAC episodes	51	5
MAC episodes	60	46

were quite different in their fit to the higher level patterns that describe overall design sequences.

The fitting of patterns to data like these involves difficult judgments about how far to elaborate them: By using more and more patterns, the fit can be made arbitrarily good. We settled on a set of patterns that gave us a sense of the sequential structure of these meetings that could be understood in terms of meaningful sequences of discussion and argumentation and that made sense in terms of design rationale (Moran & Carroll, in press).

When the 10 meetings are assessed by the fit of the patterns in Figures 9 and 10, they tend to be quite similar. However, there was at least one striking difference between the meetings drawn from the two organizations: There were much longer sequences of design discussions in company B (6.2) than in company A (3.2) meetings. Another way to state this is that the company A meetings have many more management discussions than the company B meetings, and these tend to break up the design discussions into smaller episodes. Another symptom of the greater role of management discussions in the company A meetings is that the tendency for issue discussions to be preceded by management discussions was somewhat stronger (company A = 58%, company B = 42%). This difference in the pattern of management discussion may reflect the differences in the work cultures of the two organizations. Nonetheless, the pattern of the design discussions in the meetings of the two organizations is quite similar.

Thus, the picture that emerges is that, at this higher level of abstraction, design as an activity in our 10 meetings is quite structured. It consists of episodes of design and management that are distinct from each other, and, within the design episodes, there is an orderly flow of argumentation and discussion. This is an interesting result from the perspective of both design itself as an activity and how technology might be used to support design processes. See G. M. Olson et al. (in press) for further details about the substantive implications of these analyses.

It is important to step back a moment and consider our goals. As we stressed at the beginning of our discussion of grammatical analysis, we are not interested in providing a parse of the entire sequence of coding categories for each meeting. Clearly, we could achieve this if we let our rules grow, or if we created very general categories to absorb lots of heterogeneous structure. Instead, however, we wanted to keep our grammatical rules both comprehensible and theoretically plausible and then see if we could find a sufficient number of patterns in the data. We viewed our search for comprehensible patterns in the meetings as analogous to looking for syntactic patterns in everyday speech. Although there are many regularities of pattern, there are many other factors that make the fit of a reasonable grammar less than complete.

We have illustrated how one would approach the analysis of patterns through grammatical analysis with our field data. In a subsequent paper that focuses on the analysis of sequential behavior in design, we will report the analysis of the laboratory data.

6. USE OF STATISTICAL AND GRAMMATICAL TECHNIQUES IN HCI AND CSCW

We have interleaved our presentation of the two classes of methods with examples from our own research. In this section, we present a broader range of examples to illustrate the general usefulness of the approaches we have described here. In order to make sense of the examples, we discuss sequential analysis in terms of purposes of engaging in the analysis, the heuristics for constructing descriptions of sequential structure, and the ways of evaluating these descriptions.

The purposes we examine are testing a priori hypotheses, seeking to understand the inherent structure in the data, and generating a compact description of the data. We focus on these three because they cover a large proportion of the uses of sequential analysis and serve to illustrate the interrelations among purposes, heuristics, and evaluation.

6.1. A Priori Hypothesis Testing

One might have a priori hypotheses about the sequential structure of the data, and one's purpose might be to evaluate these hypotheses. For example, one might have theoretical reason to believe that a particular expression should be quite frequent (or infrequent) in a data set. Clearly, the sequences should make sense theoretically. For example, a rational model of design discussions might lead one to expect that, after an issue is raised, one or more alternatives will be proposed to address the issue.

A relatively simple hypothesis like this can be tested directly with LSA statistics, by calculating the z score for alternative following issue at Lag 1. If it shows that this sequence occurs at levels greater than chance, the hypothesis is supported. Other kinds of hypotheses, cast for example in terms of main effects or interactions, would be testable by examining the fit of an appropriate LLM.

Such approaches are reasonable in many cases, but there may not always be good reason to require that each step in a series of rewrites be independently confirmed. The hypothesis might only require that the expression, as a whole, occur more often than expected by chance. In this case, the Monte Carlo technique would be more appropriate.

To take a specific example of hypothesis testing, the GOMS family of models provides a way of characterizing HCI data as patterns (Card, Moran, & Newell, 1983; J. R. Olson & G. M. Olson, 1990). For instance, Nilsen, Jong, J. S. Olson, and Polson (1992) were interested in method choice by spreadsheet users. The method used was characterized by sequences of actions, and the investigators were interested in how often and where certain methods were used in a stream of spreadsheet behavior. Using regular expressions or another appropriate formalism, one could represent the kinds of behavior sequences that comprise one method or another and then search behavior streams for such patterns. One could then use statistical tests to see if key sequences occur with significant frequency.

Another approach is to compare sets of data on their overall sequential organization–such as between the two conditions in our laboratory study, which we described earlier (for further examples, see reviews by Hollingshead & McGrath, in press; McLeod, 1992). One effect of using different technology conditions could be to change the sequential organization of the users' behavior. At a very general level, it is possible that behavior without technology is well organized but may become disrupted or disorganized with technology. This could show up through log-linear analysis with expected values that reflect an appropriate null condition. Second, it is possible that the behavior with and without technology might be organized differently. Such differences could show up through log-linear analysis that compares the sequential structure in the two conditions. Third, specific hypotheses about how technology might affect behavior could be assessed through LLM, LSA, or rewriting.

The effect of various technologies could also be examined by looking at sequential structure over time. One of the classic findings from the literature on expertise is that, as people acquire expertise in a domain, their behavior gets more similar to one another's (McKeithen, Reitman, Rueter, & Hirtle, 1981). Convergence on similar organization could be taken as an index of learning in computer systems.

In an especially interesting application of grammatical analysis, Smith et al. (1989) described the use of grammar-induction techniques to code automatically a stream of ongoing behavior. Smith et al. studied the use of a hypertext writing system for professional writers, WE (Writing Environment), and wanted to understand the kinds of strategies the writers used. As writers used WE, their behavior was automatically captured in time-stamped logs. These logs contained the actions performed by users, with attributes that captured the context in which an action takes place. The logs were the input to a cognitive grammar that classified these sequences of raw behaviors into various higher level coding categories. The first level of rewriting in the grammar filtered out errors, restarts, and other disfluencies. The lowest levels of the grammar code behavior in terms that are close to the software, whereas the higher levels represent classes of mental activities as defined by a cognitive theory of writing. When using this technique, Lansman et al. (1990) both eyeballed the rewritten behavioral sequences and carried out statistical analyses of them–a typical strategy for ESDA.

6.2. Seeking to Understand the Inherent Structure

There may be little theory, prior research, or experience on which one can base hypotheses, and one wants to induce structure through relatively undirected exploratory analysis of the data. Descriptions of structure are likely to be constructed iteratively, making use of many of the techniques discussed in this article. An exploratory LSA, for example, might reveal certain strong relations. These might serve as a basis for reexamining the raw data or for theoretical development. Some (or all) of these relations might be incorporated into meaningful expressions, which could be used to rewrite the data. Statistical analyses might then reveal further patterns in the rewritten data–and so on, until further attempts fail to reveal interpretable patterns.

It is essential that the investigator use exploratory statistical analysis only as input to the process of understanding and theory development, rather than automatically generating expressions based on some blind rule. Each expression should be interpretable, and, to ensure this, the investigator must remain "in the loop." The result of this sort of analysis can be evaluated in terms of its interpretability, parsimony, and face validity. The resulting expressions can be used with other data sets as hypotheses to be tested.

Entrainment provides an example of the sort of phenomenon one might want to approach in this exploratory way. The concept of entrainment in human interactive behavior means that the behavior of a person has come to be correlated with the behavior of another agent. For instance, can the sequence of behaviors of one person be predicted on the basis of another person's behavior in a social interaction, or can the behavior of a computer system be predicted in a technology–person interaction? This can be assessed by techniques such as LLM or LSA, in which we could measure statistically the dependence between two streams of behavior. This technique might be especially useful for interfaces such as direct-manipulation interfaces in which there are episodes of sustained behavior with the computer system. User characteristics such as lack of experience or computer characteristics such as poor response time would be expected to disrupt such entrainment.

Siochi and Hix (1991) searched behavioral sequences for maximal repeating patterns as a technique for heuristic evaluation of user interfaces. They reasoned that, if a particular pattern of user behaviors occurred repeatedly, then either it was an opportunity to develop a coarser grained, "macro" command or, if it contained errors, an indication of a user-interface problem. Siochi and Hix did this for both raw coding categories as well as for more abstract categories in which raw codes were rewritten into meaningful classes. Siochi and Hix found that this technique was useful for uncovering design problems with several different kinds of interfaces and stressed that theirs was a purely empirical technique for searching for recurring patterns. If one were guided by theory, by analysis of an interface design, or by prior statistical analysis, one could imagine extending this technique to more complexly defined patterns of the kind discussed earlier in the section on rewriting.

6.3. Compact Description of the Data

For some purposes, one might just be seeking as compact and accurate a description of the data as possible. In this case, as in the previous one, all the techniques mentioned in this article might be used. In fact, one would ideally like to automate the procedure, perhaps by using some form of hill climbing in which, for example, regular expressions are created to capture statistically related pairs of categories, and then each instance of the expressions is rewritten. One could iterate until there are no more statistical relations. This might be useful in some HCI applications–for example, in which one wants to represent the most frequent interactions in order to optimize their speed. For this purpose, one need not be concerned with understanding all the details of the process generating the sequences.

Siochi and Hix (1991) suggested that one could use automatically induced sequential codes as indices to the original behavior. In our own data, we could index the original videotapes by such derived categories as the MIAC code described in Figures 10 and 11 and then scan through the video records to look at episodes that the rewriting scheme had classified as being comparable. The degree of comparability could suggest whether the coding scheme is indeed capturing something that is useful or meaningful.

7. CONCLUSION

The purposes presented here are in the spirit of "ideal types," and any actual project will probably not pursue one to the complete exclusion of the others. In our own work, for example, we had a few a priori ideas about what we expected to see in the data, but we also acquired new ideas from our exploratory analyses. Despite the fact that one will often have more than one goal, considering goals separately helps one think clearly about how to proceed and is crucial when evaluating the outcome.

The repertoire of techniques available for the analysis of sequential structure in categorical data is much larger than we have been able to review here. We have tried to provide key references and have illustrated some representative techniques and the issues of application and interpretation involved in their use, especially for the HCI and CSCW contexts.

The statistical and grammatical techniques described here are, of course, only one modest part of the array of tools available for carrying out ESDA. Sanderson and Fisher (1994) review the larger collection of tools and discuss the strengths and weaknesses of the various methods, including these. Researchers who want to understand the sequential structure of their behavioral data need look at Sanderson and Fisher's excellent overview to help in deciding which methods are best for them. In our article, we have tried to illustrate the kinds of situations for which statistical and grammatical methods are useful and, further, to show how the two sets of methods can be used on concert to examine sophisticated, higher level patterns. These methods are useful when a quantitative approach is possible and desirable, but, as always, common sense must be used in applying the methods. Further, the output of these analyses is only as good as the input. The reliability and validity of the categories provide a bound on how useful the analyses can be. But, under the right circumstances, these methods can be extremely useful, both for exploring data sets and for evaluating certain kinds of hypotheses about sequential structure.

NOTES

Acknowledgments. We thank Judy Olson, Carolanne Fisher, three anonymous reviewers, and especially Penny Sanderson for their helpful comments on drafts of this article; Max Rahder for suggesting the diagrams we used in Figures 9 and 10; and J. E. Keith Smith for statistical advice.

Support. This work was supported by National Science Foundation Grant IRI-8902930 and by the Center for Strategic Technology Research (CSTaR) at Andersen Consulting.

Authors' Present Addresses. Gary M. Olson and Henry H. Rueter, Collaboratory for Research on Electronic Work, University of Michigan, 701 Tappan Street, Ann Arbor, MI 48109-1234. E-mail: gmo@umich.edu and henry@crew.umich.edu; James D. Herbsleb, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213. E-mail: jherbsle@sei.cmu.edu.

HCI Editorial Record. First manuscript received November 23, 1992. Revisions received October 8, 1993 and February 3, 1994. Accepted by Penelope M. Sanderson and Carolanne Fisher. Final manuscript received June 14, 1994. – *Editor*

REFERENCES

Abramson, H., & Dahl, V. (1989). *Logic grammars.* New York: Springer-Verlag.

Allison, P. D., & Liker, J. K. (1982). Analyzing sequential categorical data on dyadic interaction: A comment on Gottman. *Psychological Bulletin, 91,* 393–403.

Anderson, T. W., & Goodman, L. A. (1957). Statistical inference about Markov chains. *Annals of Mathematical Statistics, 28,* 89–110.

Bakeman, R., & Gottman, J. M. (1986). *Observing interaction: An introduction to sequential analysis.* New York: Cambridge University Press.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human–computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Chatfield, C. (1989). *The analysis of time series: An introduction* (4th ed.). New York: Chapman & Hall.

Chomsky, N. (1957). *Syntactic structures.* The Hague, The Netherlands: Mouton.

de Haan, G., van der Veer, G. C., & van Vliet, J. C. (1991). Formal modelling techniques in human–computer interaction. *Acta Psychologica, 78,* 27–67.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol analysis: Verbal reports as data.* Cambridge, MA: MIT Press.

Faraone, S. V., & Dorfman, D. D. (1987). Lag sequential analysis: Robust statistical methods. *Psychological Bulletin, 101,* 312–323.

Fienberg, S. E. (1980). *The analysis of cross-classified categorical data* (2d ed.). Cambridge, MA: MIT Press.

Fu, K. S. (1982). *Syntactic pattern recognition and applications.* Englewood Cliffs, NJ: Prentice-Hall.

Gazdar, G., & Mellish, C. (1989a). *Natural language processing in LISP: An introduction to computational linguistics.* Reading, MA: Addison-Wesley.

Gazdar, G., & Mellish, C. (1989b). *Natural language processing in PROLOG: An introduction to computational linguistics.* Reading, MA: Addison-Wesley.

Gonzalez, R. C., & Thomason, M. G. (1978). *Syntactic pattern recognition: An introduction.* Reading, MA: Addison-Wesley.

Gottman, J. M., & Roy, A. K. (1990). *Sequential analysis: A guide for behavioral researches.* New York: Cambridge University Press.

Grune, D., & Jacobs, C. (1990). *Parsing techniques: A practical guide.* New York: Horwood.

Harrison, M., & Thimbleby, H. (Eds.). (1990). *Formal methods in human–computer interaction.* New York: Cambridge University Press.

Hollingshead, A. B., & McGrath, J. E. (in press). The whole is less than the sum of its parts: A critical review of research on computer-assisted groups. In R. Guzzo (Ed.), *Team decision making in organizations.* San Francisco: Jossey-Bass.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika, 32,* 241–254.

Lansman, M., Smith, J. B., & Weber, I. (1990). *Using computer-generated protocols to study writers' planning strategies* (Technical Report 90-033). Chapel Hill: University of North Carolina, Department of Computer Science.

Losada, M., Sanchez, P., & Noble, E. E. (1990). Collaborative technology and group process feedback: Their impact on interactive sequences in meetings.

Proceedings of the Conference on Computer-Supported Cooperative Work, 53–64. Los Angeles: ACM.

McGuffin, L., & Olson, G. M. (1992). *ShrEdit: A shared electronic workspace* (CSMIL Technical Report 45): Ann Arbor: University of Michigan, Cognitive Science and Machine Intelligence Laboratory.

McKeithen, K. B., Reitman, J. S., Rueter, H. H., & Hirtle, S. C. (1981). Knowledge organization and skill differences in computer programmers. *Cognitive Psychology, 13,* 307–325.

McLeod, P. L. (1992). An assessment of the experimental literature on electronic support of group work. *Human–Computer Interaction, 7,* 257–280.

Miclet, L. (1986). *Structural methods in pattern recognition.* New York: Springer-Verlag.

Moran, T. P., & Carroll, J. M. (Eds.). (in press). *Design rationale: Concepts, techniques, and use.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Nilsen, E., Jong, H., Olson, J. S., & Polson, P. G. (1992). Method engineering: From data to model to practice. *Proceedings of the CHI '92 Conference on Human Factors in Computer Systems,* 313–320. New York: ACM.

Olson, G. M., Olson, J. S., Carter, M., & Storrøsten, M. (1992). Small group design meetings: An analysis of collaboration. *Human–Computer Interaction, 7,* 347–374.

Olson, G. M., Olson, J. S., Killey, L., Mack, L. A., Cornell, P., & Luchetti, R. (1992). Flexible facilities for electronic meetings. In S. Kinney, R. Bostrom, & R. Watson (Eds.), *Computer augmented teamwork: A guided tour* (pp. 183–196). New York: Van Nostrand Reinhold.

Olson, G. M., Olson, J. S., Storrøsten, M., Carter, M., Herbsleb, J., & Rueter, H. (in press). The structure of activity during design meetings. In T. P. Moran & J. M. Carroll (Eds.), *Design rationale: Concepts, techniques, and use.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Olson, J. R., & Olson, G. M. (1990). The growth of cognitive modeling in human computer interaction. *Human–Computer Interaction, 5,* 221–265.

Olson, J. S., Olson, G. M., Storrøsten, M., & Carter, M. (1992). How a group editor changes the character of a design meeting as well as its outcome. *Proceedings of the Conference on Computer Supported Cooperative Work,* 91–98. New York: ACM.

Olson, J. S., Olson, G. M., Storrøsten, M., & Carter, M. (1993). Groupwork close up: A comparison of the group design process with and without a simple group editor. *ACM Transactions on Information Systems, 11,* 321–348.

Payne, S. J., & Green, T. R. G. (1986). Task-action grammars: A model of the mental representation of task languages. *Human–Computer Interaction, 2,* 93–133.

Pentland, B. T., & Rueter, H. H. (1992, August). *Reconceptualizing organizational routines: A generative model of a technical service process.* Paper presented at the meeting of the Academy of Management, Division of Organization and Management Theory, Las Vegas.

Pereira, F. C. N., & Shieber, S. M. (1987). *Prolog and natural-language analysis* (CSLI Lecture Note 10). Menlo Park, CA: Center for the Study of Language and Information/SRI International.

Poole, M. S., & Hirokawa, R. Y. (1986). *Communication and group decision making.* New York: Sage.

Putnam, L. L. (1981). Procedural messages and small group work climates: A lag sequential analysis. In *Communication Yearbook 5* (pp. 331–350). New Brunswick, NJ: Transaction.

Sanderson, P. M., & Fisher, C. (1994). Exploratory sequential data analysis: Foundations. *Human-Computer Interaction, 9,* 251-317. [Included in this Special Issue.]

Schiele, F., & Green, T. (1990). HCI formalisms and cognitive psychology: The case of task-action grammar. In M. Harrison & H. Thimbleby (Eds.), *Formal methods in human-computer interaction* (pp. 9-62). New York: Cambridge University Press.

Shieber, S. M. (1986). *An introduction to unification-based approaches to grammar.* Stanford, CA: Center for the Study of Language and Information.

Siochi, A. C., & Hix, D. (1991). A study of computer-supported user interface evaluation using maximal repeating pattern analysis. *Proceedings of the CHI '91 Conference on Human Factors in Computer Systems,* 301-305. New York: ACM.

Smith, J. B., Rooks, M. C., & Ferguson, G. J. (1989). *A cognitive grammar for writing: Version 1.0* (Technical Report 89-011). Chapel Hill: University of North Carolina, Department of Computer Science.

van Hooff, J. A. R. A. M. (1982). Categories and sequences of behavior: Methods of description and analysis. In K. R. Scherer & P. Ekman (Eds.), *Handbook of methods in nonverbal behavior research* (pp. 362-439). New York: Cambridge University Press.

Wampold, B. E., & Margolin, G. (1982). Nonparametric strategies to test the independence of behavioral states in sequential data. *Psychological Bulletin, 92,* 755-765.