DESIGNING INTERACTIVE USER INTERFACES: DIALOG CHARTS AND AN ASSESSMENT OF THEIR USE IN SPECIFYING CONCEPTUAL MODELS OF DIALOGS

Gad Ariav
Leonard N. Stern School of Business
Department of Information Systems
New York University
40 West 4th Street
New York, NY 10012

Linda Jo Calloway
Graduate School of Business Administration
Information and Communications Systems
Fordham University
113 West 60th Street
New York, NY 10023

August 1989

Working Paper Series
STERN IS-89-105

Table of Contents

Abstract

The conceptual design of user interfaces focuses on arriving at a specification of the structure of the dialog, independent of any particular implementation approach. There is common agreement as to the importance of this activity to both IS professionals and end-users, but few -- if any -- modeling methods were developed to specifically support the *process of conceptual design*, and the usefulness of such methods has not been adequately addressed. This paper introduces the **Dialog Charts** (DCs), and documents a preliminary examination of their perceived usefulness by designers of user/system interaction who actually used them. The DCs yield high level dialog schemas that are abstract enough to support the conceptual design of dialog control structures. In a uniform diagraming framework they combine the concept of dialog independence, distinguish between the dialog parties, provide for hierarchical decomposition and enforce a structured control flow. The usefulness of the DCs has been studied empirically in a qualitative inquiry. Recalled experiences of designers were captured and analyzed to ascertain the concept of usability, as well as assess the usability of the DCs. Usability has emerged from this study as a set of 38 concerns that operationalizes the broader aspects of purpose of use, design stage, impact on product structure, impact on design process, and attitudinal patterns. In general, the Dialog Charts were found by these dialog designers to be a useful, exhibiting the essential attributes of tools for conceptual modeling.

Categories and Subject Descriptors:
D.2.1 [**Software Engineering**]: Requirements/Specifications--*methodologies*; D.2.2 [**Software Engineering**]: Tools and Techniques--*user interfaces*; I.3.6 [**Computer Graphics**]: Methodology and Techniques; H.1.2 [**Information Systems**]: User/Machine Systems--*design*

General Terms: Conceptual Design, Evaluation

Additional Key Words and Phrases: Dialog models, human-computer interaction, qualitative research

1. Introduction: The Management of User/System Dialogs

The intensifying discussion of conceptual dialog models is a direct result of the recent consolidation of a widely supported dialog management framework. This framework partitions system/user dialog into three linked generic functions: the handling of syntax, the handling of control and the handling of the applications (Figure 1-1). This conceptualization essentially underlies a wide array of contemporary views of dialog management, expressed in various terminologies (e.g., [34], [3], [31], [16], [1], [35], [17] [32], [13] and [15]). The set of dialog concerns is parcelled out as follows: the *syntax* defines the valid set of user inputs ("gestures"), and captures presentation aspects, including the delivery of outputs to the user; the handling of the *applications* entails the definition of the interface to the required application modules and the passing of information to and from these modules; finally, the *control* aspect of dialog management is concerned with the maintenance and enforcement of the **dialog structure,** practically defining the set of interaction contexts and the permissible sequences of user-system activities.

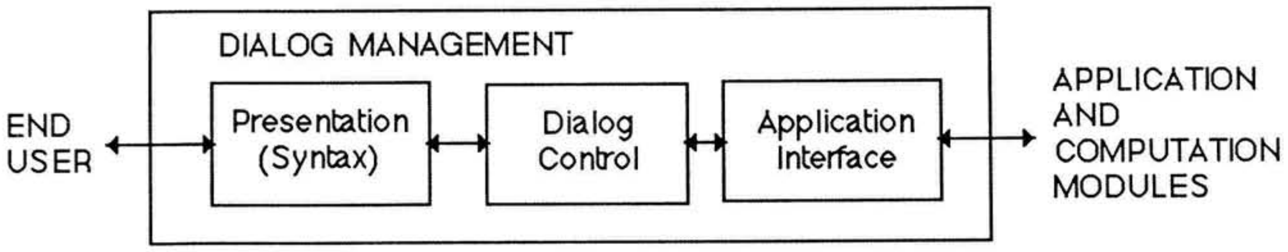


Figure 1-1: The Generic Structure of Dialog Management

This tripartite view of dialog management implies that essentially, the dialog structure of a system can be designed independently. A model of the control structure of the dialog is, therefore, a stable abstraction of the dialog: it outlines possible sequences of system/user interactions without being bound to the specific implementation details like the interaction style or implemented appearance of the user interface. Following this logic, the control structure of the dialog corresponds most closely to what might be considered to be the dialog's conceptual model. Such a conceptual model captures a structure of the dialog that is as close as possible to the user's functional view of the task or tasks. The model is then *translated* concurrently into more detailed and formal implementation models of both the syntax and the application, which are translated in turn into a concrete physical model.

Generally, there seem to be three typical approaches for designing dialog structures; those which focus on a correct design process, those which prescribe a proper set of dialog attributes, and finally those which provide tools for dialog modeling and analysis. *Procedural approaches* describe sequences of activities that dialog designers should follow. These approaches sometimes use formal or informal representations, but the emphasis is on how to approach the design and on how to decompose the task (e.g., [7], and [26]). *Guidelines sets* are loose collections of principles, policies and rules to be used in dialog design (e.g., [42], [12], [29], [30], [36] and [38]). A "guideline" advises about the proper conduct for the dialog; for instance, "Control should always remain with the user." *Analytic methods* employ an abstract and somewhat formal representations of the interaction, along with rules for manipulating these representations. The Dialog Charts, the topic of this paper, belong primarily to this last category.

The general state of the art of conceptual modeling of dialogs is rather problematic: "While there is nearly universal agreement that [conceptual design] is the most critical point in the process, there is also a nearly universal lack of adequate tools and formalisms to aid the designer at that task" (p.314 in [33]). Jim Miller in [37] further identifies the support for the *process of design* as a "real bottleneck" in the areas of interface design and development: "If the role of interfaces is to help users understand and work with the semantics of a task domain, we need tools that will let interface designers represent these domains and make their important properties explicit in the interface." (p.199).

This paper presents and examines an approach for the specification of this type of dialog model. Although several methods have been suggested for modeling and specifying human/computer interactions, they are often oriented towards programmers. These methods typically address implementation aspects of dialog design, and furthermore, they do not directly support the *process* of dialog design. The conceptual modeling of dialogs and the Dialog Charts directly address these concerns.

The Dialog Charts (abbreviated henceforth as "DCs"), are introduced in Section 2, together with some discussion of the nature of conceptual modeling of dialogs. A manifestation of the early formative stage which characterizes the area of conceptual modeling of dialogs is the lack of commonly accepted criteria for assessing the impact of a modeling approach. Section 3 outlines a qualitative and comprehensive research methodology which aims at revealing salient aspects of "usefulness" in actual use of conceptual modeling tools. Section 4 documents the pilot application of this approach in an empirical investigation of usefulness of the DCs, and summarizes its findings. The discussion in Section 5 highlights the main results of this study, and puts them in somewhat broader perspective.

2. The DCs Approach to the Conceptual Modeling of Dialogs

A tool for conceptual modeling of an application or an aspect of it, like the Dialog Charts, is fundamentally a method for solving high level design problems. It provides a medium for translating a set of informal user requirements into more concrete and specific constructs that further guide the development of the application [21].

In developing the elusive notion of conceptual models of dialogs, analogous concerns in the area of database design provide some useful insights. The contemporary view of database design clearly differentiates among three tiers of models [40]: the conceptual model, the implementation model, and the physical model. The database design process is therefore viewed as the gradual refinement of system specifications through the development of a consistent set of corresponding models (Figure 2-1). Conceptual models (e.g., Entity Relationship Model), capture users' views and outline fundamental system requirements; these models are ideally expressed in ways which are directly examinable by users. Implementation models (e.g., Network Model) add formality and precision within the framework established by the conceptual model. Moving closer to the realm of computing, physical models further ascertain the feasibility of the system by translating the implementation model into concrete data and software structures, relating them to available hardware options.

The analogy, it seems, can form a useful agenda for the discussion of proposed methods for constructing conceptual dialog models. By far the most influential conceptual data model is Chen's *Entity/Relationship Model* (ERM) [6]. Date's critical remarks concerning this model

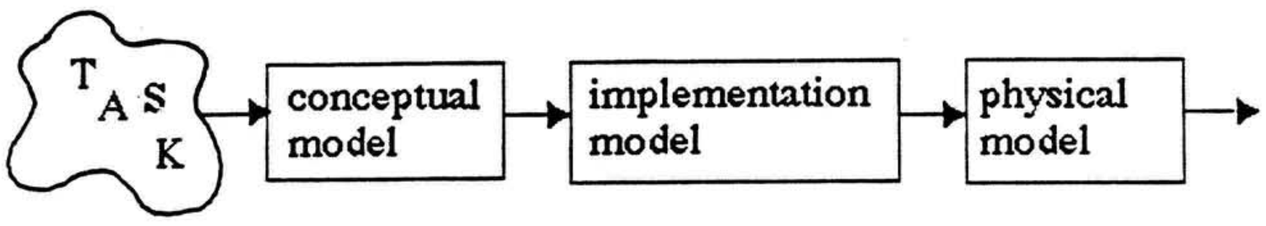


Figure 2-1: Model Hierarchy in Database Design Processes

(pp.611-612 in [8]) are of particular interest. Specifically, ERM is said to be vague, imprecise, loose and not well-defined; its definition may not meet all the requirement considered necessary to qualify as "true" data model; it is said to be a "thin layer on top" of the much more rigorous relational data model; that it leaves crucial modeling aspects implicit; and that its popularity could be attributed to the diagramming technique. One can argue that it is precisely these deficiencies that make the ERM so useful as a method for conceptual design: They directly correspond to the quintessential attributes of the early stages of the analysis and design of database applications.

The Dialog Charts discussed in this paper were similarly conceived to facilitate the early stages in the design of dialog structures, and critiques like the above can be rightly leveled at them. Nevertheless, the charts seem to provide an effective vocabulary for the specification of conceptual dialog models and for solving dialog design problems.

2.1. The Dialog Charts Notation

The principles that underlie the Dialog Charts reflect the variety of concerns relevant to the design of user/system interaction. Specifically, the concepts formulated as the framework for the Command Language Grammar [28] are used in the DCs to identify the structural elements of human/computer interactions. The design discourse assumed and supported by the DCs is made of cycles among the basic design activities of *goal elaboration*, *design generation* and *design evaluation*, until a satisfactory specification is found [23]. Finally, the types of control flows in the DCs and their diagrammatic nature correspond to some key notions of the Syntax Charts [20].

Six distinct constructs make up the DC notation. They are associated with a set of graphic symbols (Figure 2-2).

1. A decomposable user activity, i.e., a composite gesture (indicated by a box).
2. A non-decomposable user activity, i.e., "terminal" (an oval).
3. A decomposable system activity, i.e., a program (a double box).
4. A non-decomposable system activity, i.e., a reasonably "closed" and well-defined subroutine (a double oval).
5. An activity that combines user activities and system activities, i.e., a task or a method that involves user and system interaction. Such tasks could be either user-led or system-led (indicated by different combinations of a half single, half double box).
6. Direction of flow (indicated by an arrow). The basic flows permissible are selection, iteration, sequence and case. These can be combined arbitrarily.

The arrows represent the directions of the sequences, and thereby play a critical role in the DCs' capacity to explicate structure. By limiting the repertory of flows to those commonly associated with structured programming approaches, a measure of desired quality is enforced on the result of the design. Specifically, structured flows can aid in identifying robust dialog logic and modular dialog design. Similar arguments have motivated the inclusion of these constructs in lower levels of dialog modeling (e.g., [2]). Junctions in the diagrams represent decision points, and are resolved by whomever holds the initiative at that point. The party (i.e., either user or system) whose range of actions is specified in the routes that branch out of the junction holds the dialog initiative and selects the actual dialog path to be followed. This approach requires the adherence to a consistency constraint, namely that all the paths that emanate from a junction will be either all user-led or all system-led. It also brings out the fundamental decision on the assignment of dialog initiative, and explicitly calls for its resolution.

The DCs focus on "holistic" description of the dialog, following a "top-down" design process. The major manipulation in dialog charting is the transformation of an element in a dialog chart into a detailed chart. More formally, a *transformation* is a gradual or marginal modification of a consistent set of schemas into another consistent set of schemas. Specialization and generalization transformations [11] correspond to the refinement of DC "boxes" into their underlying elements and regrouping DC elements into an aggregate dialog element, respectively. The range of these manipulations and the associated rules have been kept intentionally limited, to preserve simplicity. The rules are that any box can be further decomposed. It can be

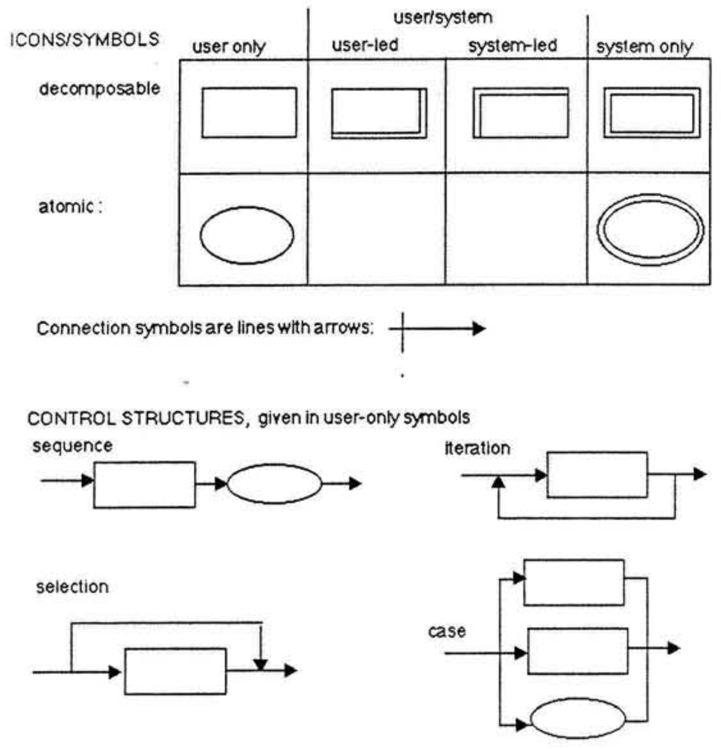


Figure 2-2: Dialog Charts notations and icons

decomposed into more boxes or into boxes and ovals, or into ovals. However, once a box is either "all user" (i.e., single-lined box), or "all system" (i.e., double-lined box), it can only be decomposed into more boxes and ovals of the same kind. Ovals are atomic and can't be further decomposed. An additional restriction -- the consistent initiative constraint -- applies to the choice of the first ("left-most") element in the decomposition of user/system activities or "mixed party" tasks: This leading element has to reflect the definition of the original task as either user-led or system-led. For instance, the decomposition of a user-led task should start with only user activities or user-led mixed-party tasks.

A classical issue in design, and especially in conceptual design, is how deeply should the structure be decomposed. A related issue is when an element is declared as a terminal rather than as a further decomposable. There is no clear stopping rule for the elaboration process. Design common sense, however, indicates that the process should stop either when further decomposition does not offer new relevant insights, or when the element represents a library module or "canned" procedure.

The specification of error handling procedures within the structure may tend to obfuscate designers' and programmers' views of the underlying structure. To avoid this confusion, the DCs generally follow the notions embedded in the Syntax Charts of Jensen and Wirth and support the concept of designing only the permissible dialogs [20]. Error handling procedures are added to the DCs as annotations at the appropriate system level.

The DCs focus exclusively on conceptual dialog modeling, and address its essential aspects by integrating simple visual concepts, structured flows, hierarchical decomposition and distinguishable dialog parties. According to [11] the purpose of conceptual models of user interfaces is to provide (1) an abstract representation of dialogs (i.e., be a basis for a *set* of functionally equivalent interface implementations), (2) a specification for the development of user interface software, (3) a means to ascertain correctness and completeness, (4) a means to evaluate the design with respect to speed of use and ease of learning, and (5) a run time help to the user. The DCs seem to address all these concerns, with a lesser emphasis on the last.

While no single tenet of the DCs is in itself novel (as clearly indicated by the citations earlier in this section), their **integration** in the context of dialog design is. The Charts were initially developed in 1982 and were used since then in numerous projects of system development where interactive decision support systems and online database systems were designed. The DCs are typically taught and demonstrated in about an hour of formal instruction, during which sufficient proficiency is gained.

2.2. DC Models of Dialog Situations

The two examples in this section demonstrate the use of the DCs in the design of new dialogs or the analysis of existing ones. First, the DCs are applied to the conceptual design of a LOGIN command in a Military Message System (Section 6.4 in [19]). In a second example the DCs are used to model and describe the structure of basic dialog of the popular Lotus 1-2-3 product. For demonstration purposes this section focuses on simple examples.

In the LOGIN task a user enters into a dialog with a computer in order to establish a session [19]. The specific scenario is as follows: The user enters his or her name. If the system doesn't recognize the name, the user is prompted to try again. When the user enters a valid name, the system prompts for a password. The user gets two tries to enter a correct password and proceed. If an incorrect password is entered twice, the user must begin the whole command again. On receipt of a correct password, the user must select a security level for the session, which must be no higher than the user's security clearance. "If he enters a level that is too high, he is prompted to reenter it, until he enters an appropriate level. If he does not enter an appropriate security level, he is given the default level unclassified." (p.44 in [19]). Note that the specification is somewhat ambiguous with respect to the dialog logic -- there are two consequences of entering an inappropriate security level.

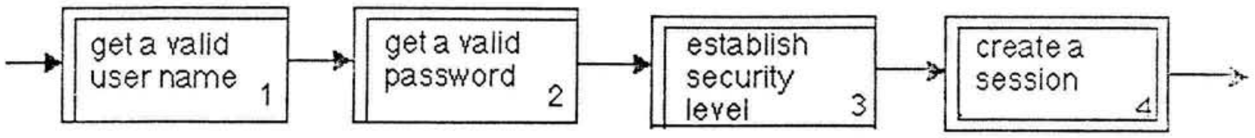


Figure 2-3: MMS LOGIN Session, Topmost level DC

The DCs for this scenario are provided in Figures 2-3 through 2-5, with each figure representing a different level of system elaboration. Figure 2-3 represents the topmost view of the session. It allows the designer to partition clearly the overall flow into well defined concerns. In some cases, the first level of elaboration may be enough. However, in order to gain more insight into the LOGIN procedure a further decomposition should be worked out. Figure 2-4 includes two successive levels of elaboration for the box numbered 2 in Figure 2-3. In another example, Figure 2-5 represents a second and third level "explosion" of the box numbered 3 in Figure 2-3.

Note how the use of the structured DCs forces the designer to disambiguate the verbal description of the session. In the DC, the interpretation is explicit: The user is either allowed to indicate no security clearance, or is allowed to enter a valid security clearance level.

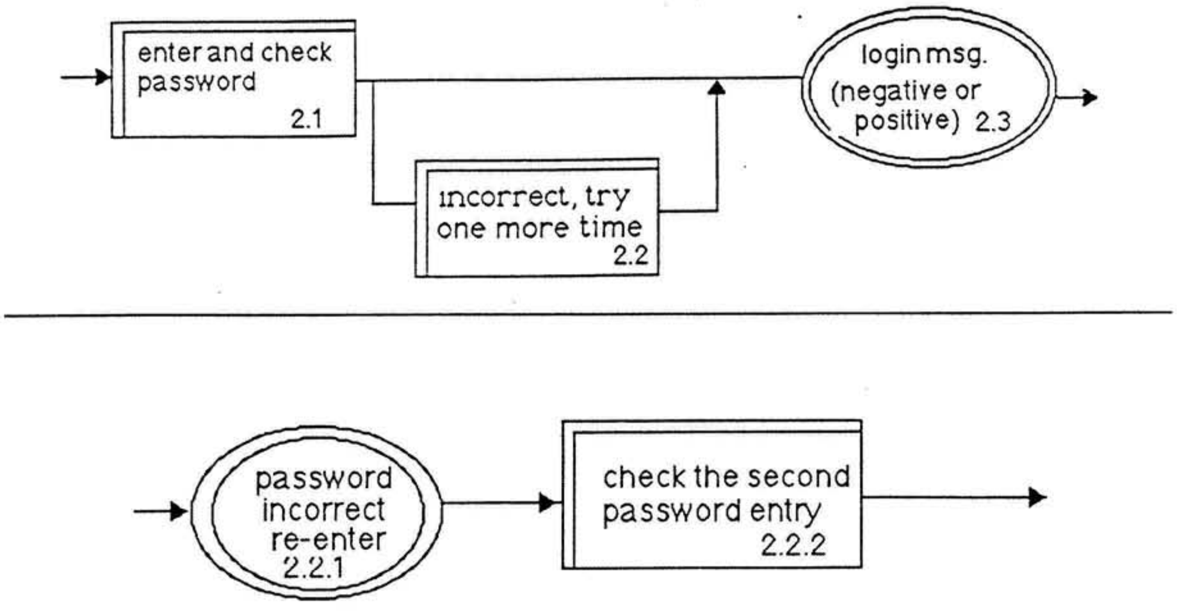
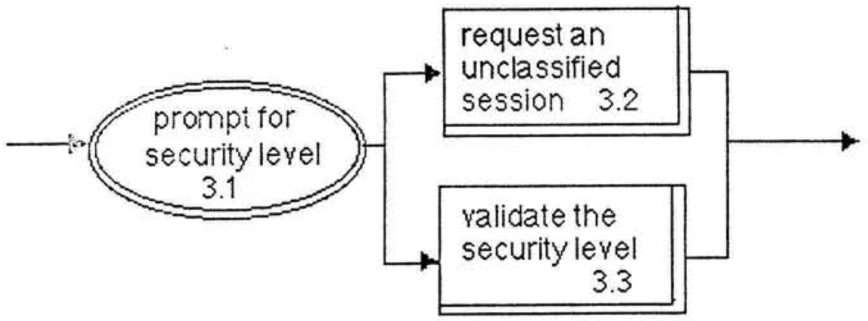

enter and check password 2.1
incorrect, try one more time 2.2
login msg. (negative or positive) 2.3
password incorrect re-enter 2.2.1
check the second password entry 2.2.2

Figure 2-4: Levels 2 and 3 DCs for "Password Getting" subtask

The complete set of DCs for the LOGIN session shows which system modules and subroutines need to be programmed, those in double-lined symbols. The collection of the double boxes and ovals therefore serves as a preliminary blueprint for the detailed design of the applications and the application processor. If, however, all double boxes and ovals are removed from the charts, the remaining set of connected user actions (i.e., the single-lined elements) constitutes a broad definition of the user interface syntax, as it practically identifies the complete valid user-generated syntax.

A (partial) description and analysis of the popular spreadsheet package 1-2-3 (by Lotus Development Corporation) is conducted in Figures 2-6 through 2-8. In Figure 2-6 the top level of interaction is specified, indicating clearly the extent of choices available to the user. Figure 2-7 is an explosion of the user-led task labeled **Commands** in Figure 2-6, highlighting the choices available to the user at that stage. Figure 2-8 further elaborates on the structure of the function **Copy** that has been offered to the user at the **Commands** level dialog.

This analysis of an existing dialog highlights some interesting observations about the DCs.


prompt for security level 3.1
request an unclassified session 3.2
validate the security level 3.3

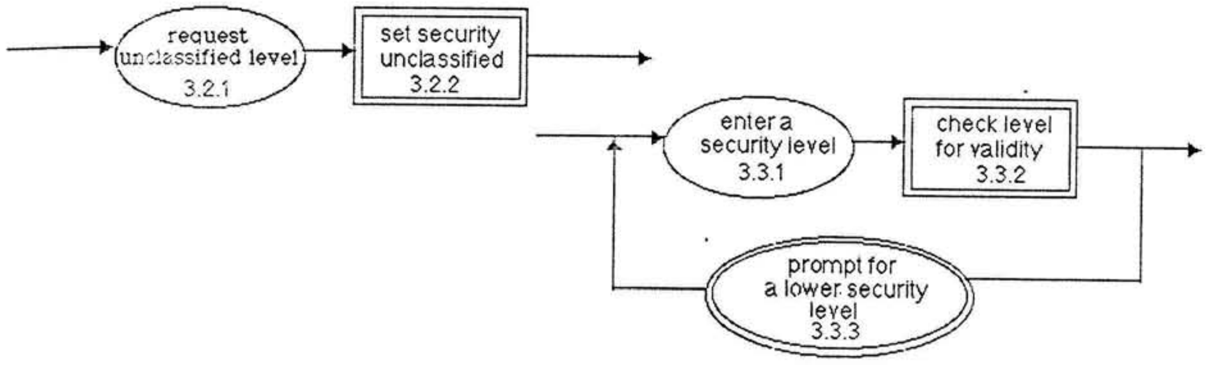

request unclassified level 3.2.1
set security unclassified 3.2.2
enter a security level 3.3.1
check level for validity 3.3.2
prompt for a lower security level 3.3.3

Figure 2-5: Levels 2 and 3 DCs for "Security Establishing" subtask

As far as the explication of the extent of control goes, the three figures are visibly different -- the taller the figure, the looser is the structure, and the user has to confront a wider set of choices. This in itself is neither "good" nor "bad", but rather indicates instances in the design where tradeoffs between freedom and confusion should be evaluated. The structure of the **Copy** command is markedly different from the other two -- it is closer to a linear, tightly controlled sequence, with relatively limited extent of user choices in carrying out the task involved. The DCs also render explicit the lack of "structuredness" in the sequence of activities that leads to quitting the session (the "extra" exit from the bottom box in Figures 2-6 and 2-7). Again, the DCs bring the unstructured sequences to the designers attention, adding it to the design agenda. The final decision whether to retain that structure or "correct" it is a question the designer has to ultimately decide.

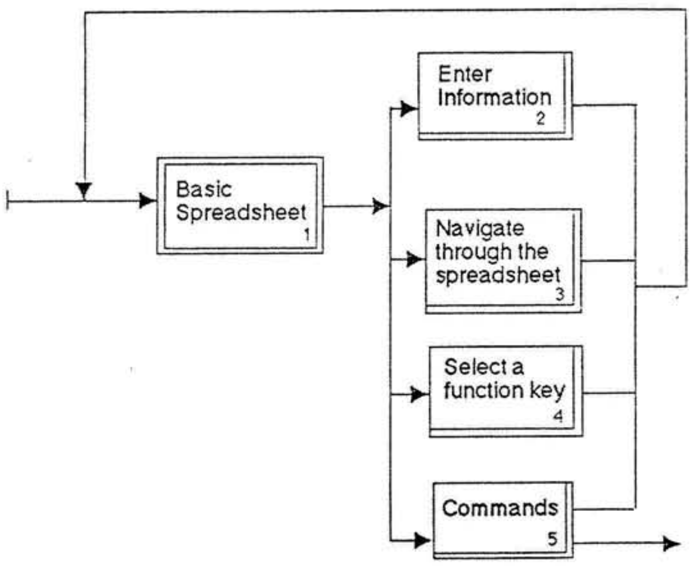


Figure 2-6: DCs for Top Level Lotus 1-2-3 Dialog

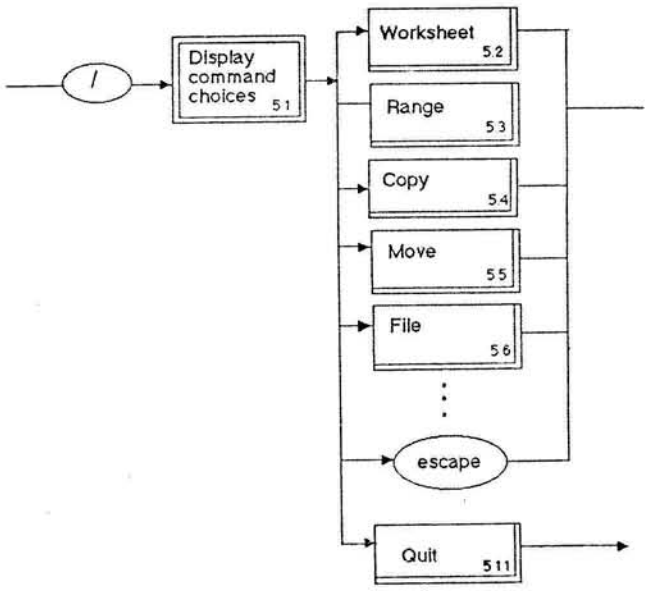


Figure 2-7: DCs for Lotus 1-2-3 **Command** Dialog

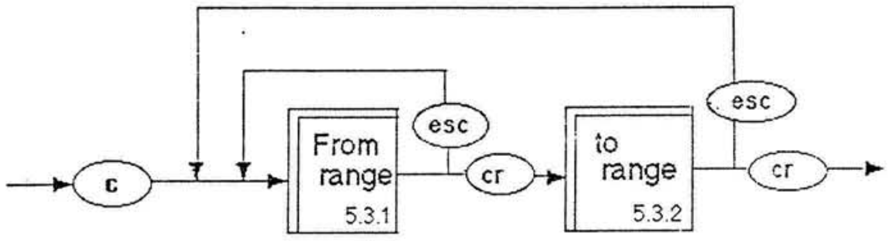


Figure 2-8: DCs for Lotus 1-2-3 **Copy** Dialog

Another comment relates to the wide range of implementation possibilities addressed with a DCs-based model. As it turns out, each of the three dialog models in Figures 2-6 through 2-8 is implemented in a different interaction style: the top-level dialog is implemented as an unprompted interaction, the **Command** follows primarily a menu-style interaction (or as a pull-down menu in some 1-2-3 "clones"), with an alternate unprompted and abbreviated style, while the **Copy** command is implemented in a Question/Answer style, with direct manipulation being an optional type of user's gestures. The actual decision about interaction style is probably affected somewhat by the fundamental properties of the dialog as they are picked-up by the DCs, but the determining factor is a set of assumptions about the user. Otherwise, the sharp difference between the implementation of the top-level and the **Command** dialogs cannot be easily explained.

3. Studying The Usefulness of a Conceptual Design Tool

How dialog designers actually use conceptual design tools? Apparently, we know very little about it: "Most people who have built tools for interface development claim that these tools enhance designer performance. The authors are not aware of any empirical evidence to support these claims" (p.233 in [18]). There is, therefore, a need to address first the issue of how the use of conceptual design tools can be or should be studied. In this section we outline a corresponding research approach, and highlight the deliberations that underlie it.

There are two basic premises. The primary is that to be "useful" to the designer, a tool

- **Q1. Purpose**
  Gathering intelligence, goal elaboration, design generation, design evaluation, communication
- **Q2. Stage**
  Documentation and analysis, logical/conceptual design, implementation design, programming/coding, testing
- **Q3. Product**
  Hierarchical and modularity structure, control structure, data structure/architecture
- **Q4. Process**
  Design philosophy, constraints
- **Q5. Attitudinal Patterns**
  Learning, task performance, subjective satisfaction, retention, errors

Figure 3-1: The Original Seed Categories

The overall research strategy adopted here can be classified as "qualitative." In seeking broad but valid responses to the above questions our approach draws primarily on concepts of grounded theory [14], [24], qualitative analysis methods [27], and qualitative content analysis [22]. In Section 4 we present the results of applying this methodology in studying a team of designers who had just concluded a system development project in which they used the Dialog Charts.

The experimental setting for data collection is a field experiment [25], and the experimental task is the analysis, design, development and demonstration of an interactive computer-based application. It is assumed that the participants have reviewed various methodologies for disciplined design of information systems and their components (e.g., databases and user interfaces).

As indicated above, the main objective while capturing the data is to solicit designers perceptions of the DCs in an unobtrusive fashion. Following the completion of the development of their system, the designers participate in an open-ended, semi-structured and funneled interview with a hidden-agenda [5], [10], [39]. In such an interview questions are prespecified, but the answers are not, and the broad range of.questions masks the identity of the actual topic under study. The funneled interview begins by asking questions about a general area or domain, and then pursues areas that have been mentioned by the interviewed team more specifically. The results of this approach are that each issue on the interview's hidden agenda is approached with the broadest and most open questions first. These are followed by more specific questions, often rephrased according to the specific language used by the informants.

The first segment of the interview establishes the overall context (i.e., What does the system do?), and then focuses on the work that the team accomplished in the various stages of system development -- from conceptual design all the way to actual coding. The second segment of the interview raises two issues. It starts with a discussion of the problems encountered in specifying, designing and implementing the dialog. It then brings up the topic of design tools, and future intentions regarding tools that have been used. Throughout the interview no direct focus is placed on the tool under study (the DCs in this research), in order to preserve the hidden agenda, and to guarantee that information about how the designer used it is, to the extent possible, voluntary. The audio-taped interviews provide the raw data for analysis.

Basically, qualitative analysis consists of progressively reducing and categorizing raw data into various forms of *display*, i.e., "an organized assembly of information that permits conclusion drawing" [27]. The analysis is an iterative process of data reduction, display, and conclusion refinement. In this way, the data which at first seems vague and inchoate gradually becomes more explicit and "grounded" [14]. Initial data reduction of the taped interview is achieved through a structured content analysis; i.e., the tracking, extraction, transcription and categorization of explicit "mentions" of the tool. A *mention*, hence the basic unit of analysis in this study, is a group of utterances made by the designers about the tool, within a design context and categorization. A change in the broad context or major category signals the end of the mention. Mentions occurred in *sequences*, i.e., one or more mentions that are contiguous.

The mentions are then encoded by studying their relationship to the concept of usefulness, as operationalized by the set of five research concerns and their corresponding topical categories, as listed earlier. The encoded mention frames either fit in any of the existing categories or a new category is declared. In the process, the initial categories could be partitioned or combined, and new ones could be added, as new properties and value sets are noted as suggested by the mentions that are encountered during data collection, reduction and analysis [27]. A mention can relate to more than one question or category. Mentions are then studied to establish data-context relationships, where the mentions are the independent variable and the context targets appear as the dependent variables [22]. By grouping mentions according to their categories, tallying them and comparing their contents, the general categories assume specific, concrete, and grounded meaning. By studying mention frequencies, design concerns can be sorted out, reaffirmed or refuted.

The basic assumption is that mentions, because of the unobtrusive data collection approach, faithfully represent and reveal the *perceptions* of the designer. Making inferences from mentions about *actual* use is problematic -- the linkage between mentions of use and actual usage is not directly observable. For example, it is possible that some users will not voluntarily mention using the tool. In this case we assume that although the tool was used, it is unlikely that it is perceived as either useful or as significant part of the development process.

The empirical research reported in the next section is an analysis of a single site, a case study of sorts, which tries to ascertain the feasibility of the research method, as well as develop initial appreciation for the usefulness of the DCs. Even though a case study is scientifically "weaker", it is nevertheless rich and unconstrained, as befits the preliminary state of understanding of the usefulness of dialog design tools and the processes of dialog design. In particular we attempted to determine whether or not the initial seed categories appeared to represent the ways the designer used the Dialog Charts, and whether the basic questions were sufficient to cover the broad areas where the Dialog Charts were mentioned.

4. The Usefulness of the DCs

This section summarizes the pilot application of the approach outlined in Section 3 to explore the usefulness of the Dialog Charts in conducting system design activities. The DCs were well received by their users in varied design situations and a wide range of applications in literally hundreds of systems developed in and outside academia. Nevertheless, there was no methodical basis for substantiating this anecdotal evidence, or for identifying the reasons for the popularity of the DCs.

Data were gathered in a field experiment, which occurred over a period of about three months. The experimental task was the design and development of an interactive database application. The application's scope, complexity and development mode were realistic -- a team-based development setting of a system of about 1000 lines of high-level code. The team was made up of 4 undergraduate students in their senior year, all Information Systems majors, who were enrolled in a course on the analysis and design of interactive systems. The interviewer was an outsider who did not participate in any of the previous stages of the experiment itself.

Participants' inexperience (relative to practicing information systems professionals) does not in itself limit the generalizability of the results. In the era of end-user computing many designers of interactive systems, especially those engaged in the early stages of system definition and task specifications, are probably less thoroughly trained in systems design. As it turned out, most of the participants took up jobs that required them to participate immediately in designing interactive systems.

To highlight the nature of the qualitative data analysis, the summary of the findings is presented in three complementary fashions. Following a brief discussion of the broad distribution of mentions into categories, we consider the observations category by category. Finally, we comment about the observed relationships among the categories indicated by the data.

4.1. Distribution of Codings

In all, there were 49 mentions of the Dialog Charts throughout the 90 minutes long interview. The 49 mentions received 80 category codings. We considered a mention as "reliably coded" if there was no disagreement about the applicability of the codings, although there might be other codings that could apply.

Figure 4-1 summarizes the mentions of the DCs by the team and underscores the richness of the information that is under study.

By far, the majority of the coded mentions related to **purpose** and **attitudinal patterns**. With respect to the **purpose** of use category, Dialog Charts were mentioned most frequently in the context of *design generation*. While this could be expected, somewhat unexpected was the intensity of mentioning the use of the tool for the *communication* of design information. Although communicating has been noted as a characteristic of a usable development methodology [41], there was no requirement that the team use DCs as a communications vehicle.

The large number of mention categorizations under **attitudinal patterns** came as a

	totals by sub-category
Purpose	
Intelligence	
Goal elaboration	
Goals into sub-goals	2
Design generation	
DCs as step before pseudocode	2
Queries and prompts	1
Menus	3
Designing the "user interface"	2
User orientation in interface design	4
Error processing	2
Coding	1
Design evaluation	
For completeness	1
Menu structures	2
Flow of control	1
Communication	
To later stages	2
Logical to physical	4
Logical to coding	2
Logical to maintenance	1
Stage	
Documentation and analysis	
Logical or Conceptual design	
Physical/Implementation design	1
coding	2
Testing	
*Maintainance	1
Products structures	
General modular/hierarchical	2
DCs as a map, diagram	2
*User orientation to product	2
Control structure	2
Data structure/data flow	2
Process of design	
Constraints on design	
Thoughts forced into something concrete	1
Help maintain strong control	1
Philosophy of design	
Distinguish the parties	2
Top-down decomposition	1
Iteration	1
Attitudinal patterns	
Learning	
Dialog Charts/should have learned value earlier...	6
Recall/retention	2
Task performance	
Time to do menus/little	1
Time to do DCs/lots, because "had" to re-do them	1
*Task comprehensibility	
Saw all levels of depth	1
Saw control structure	2
Subjective satisfaction/dissatisfaction	
Intend to use DCs	4
Surprised, because tool useful	4
Adjectives valuable, important, etc.	8
Confident in code	1

Figure 4-1: Tabulation of Mention Codings

surprise: *subjective satisfaction* was the context for 17 of the codings. Eight of these referred to the DCs as *valuable and important*, while 4 referred to the *surprise* of the team by the usefulness of DCs.

No major categories had to be added, but some remained "unmentioned." Since this is only the first of a series of sites to be studied, no categories were formally dropped at this time. For instance, there was no mention of the use of the DCs for the purpose of *intelligence gathering*. A few secondary categories emerged (they are marked with an asterisk in Figure 4-1). For instance, the DCs were mentioned in connection to *system maintenance*, a system development **stage** that was originally thought to be removed from a conceptual design tool. Designing with a *User orientation* was identified as a characteristic of dialog design **products**, and *task clarity and comprehensibility* was an aspect of **attitudinal patterns** that did not seem to be captured by the original seed categories. All third-level categorizations were suggested by key words in the empirical data. They basically refine their corresponding categories and give them a more precise and concrete interpretation.

As stated in Section 3 above, the second segment of the interview focused more directly on dialog design. Counter to our expectations, the extent of mentioning the DCs in the two segments of the interview was similar. The total number in the first segment was 27 mentions and 44 codings, in 11 sequences. During the second segment, in which somewhat more direct questions were posed, there were 22 mentions in 12 sequences, which were coded into 35 categories. Figure 4-2 shows the mention breakdown by segment.

One interpretation is that the team has formed fairly stable opinion about the DCs, and therefore related to them consistently across the different modes of evidence gathering.

4.2. Category by Category Summary of Mentions

In the following paragraphs, the content of mentions in each category is summarized and illustrated by examples. The mentions are reproduced in their entirety in [4], and the numbers in the parentheses following mention quotations refer to the mention's chronological sequence number.

Purpose. The categories mentioned under purpose were *Goal elaboration, Design generation, Design evaluation* and *Communication.*

	Sequences	Mentions	Codings
Segment I	11	27	44
Segment II	12	22	36
	23	49	80

Figure 4-2: Tally of sequences, mentions and codes by interview segment

Goal elaboration includes decomposing goals into potential sub-solutions or subgoals. They described using DCs to "differentiate between system -- response or function, and user response or function, or something that's a combination of both" *(29)*. The products of *goal elaboration* are the functional requirements of the system. This design process and its results are as paraphrased in the following mention:

- ...You start out with the very simplest, the highest level... break that down, and you go down and down and down until you hit the lowest level. You hit every possible situation.
- You can't explode anymore.
- Until you don't need to prompt the user for anymore information.
- ...and you can just perform the necessary functions. *(30)*

Design generation was the most frequent context in which the team mentioned the DCs. The DCs were mentioned in the context of designing queries and prompts, menus, the "user interface", the control structure, the code, and error handling. The DCs are a conceptual design tool, and it was a surprise that the team used them at the most detailed level of design--designing the code. Perhaps they might have used another tool vocabulary if one had been available to them, but they did not express any feelings that they felt the need of a design vocabulary more specifically targeted towards implementation design or physical design (coding).

Early on the team mentioned designing *queries and prompts* in conjunction with a *user orientation* towards eliciting necessary details from a user during system operation:

> - ...you have to set up queries, or question/answer things, whether its menus or whatever. Somehow, the system has to egg-on the user, know what I mean? Lead them into what the system means.
> - How the system will prompt you into getting to what you want.
> - That's the whole idea behind the Dialog Charts. *(3)*

This context of *user orientation* surfaced elsewhere in the interview, particularly in reference to the *design generation* of the dialog. In response to the question "how did you go about specifying and designing your interface?" the team answered,

> - How you would most feel comfortable if you would put yourself in the user role.
> - That basically came out of the Dialog Charts too... *(28)*

One recurring theme in the interview was the surprise expressed at the usefulness of the DCs. It surfaced while discussing *menu* design, in the logical design phase of the interview: "We really did use them [the DCs] as far as designing menus" *(12)* and "The menus really came out of that [the DCs]" *(25)*.

It is interesting to note that the team extended the DC design vocabulary to designing *error processing*. One mention in particular described how the team had integrated error processing with general control structure design. As indicated in Section 2 above, the DCs avoid cluttering the description of the dialog with its entire collection of alternative paths of error handling. The team further noted:

> - ...I don't know whether it's supposed to include it or not, but we included the error, because -- well I don't know, to be quite honest with you, but we got very familiar with it [dialog charting]...I mean we just thought that it was just a logical extension of it. *(43)*

Apparently the team felt free to <u>change</u> the tool to suit their purposes, which indicates the team's familiarity and comfort with the DCs.

Finally, the team mentioned using the DCs, among other design products, in *coding* their system:

> - ...We took our Dialog Charts, and our files, and our menus that we designed, and we... actually started to code them. Coded the record layouts, coded the file description statements.
> - Set up the user interface.

- Coded the menus, yes. *(27)*

The mentions of the DCs in the context of <u>*design evaluation*</u> seem to reemphasize the familiarity that the team found with the charts, because they could use them in a flexible fashion, and even comment on suggested improvements to the design process. They mentioned using Dialog Charting in an iterative fashion to evaluate their designs for completeness:

- ...What we found was the Dialog Charts really needed to be an interactive process. Because as you go through them and through them and through them --
- You realize things that you haven't thought about before...*(20)*

The team also commented that DCs were used to reevaluate their menu hierarchy. For example, one comment was, "...we had originally gone back to the original menu, and then decided that that's boring" *(34)*. They were apparently satisfied with their restructuring, because as the mention continues, they note that the control structure of the system had become "more flexible" and "efficient".

The DCs were often mentioned in the context of <u>*communication*</u> from task to task. The DCs were used to derive menus, as input to the coding phase, and in determining how to prompt the user. Succinctly: "They really gave us a basis for so many of the next steps." *(37)*. As one member commented, when there was a question of the value of the different design tools (i.e., dataflow diagrams, flow charts and dialog charts), the DCs "seemed to be the most helpful... because when we did get into the later stages, we did actually use them. Much to my surprise." *(10)*. Similar comments were repeated later in the conversation.

Interestingly, one team member, while indicating his intention to use the DCs in the future, focused on using them to communicate with users and in system maintenance:

-... In terms of helping them maintain their system, I do keep in the back of my mind the Dialog Charts, which I thought were great. In terms of helping explain myself to them, what ideas I had. Whereas before maybe it was just kind of haphazard. Now I have some structure for explaining, and why I'm thinking what I'm thinking. *(48)*

Stage. References in this category were scant. Nevertheless, three of the four mentions related to <u>communicating</u> information among the various system development stages; for example, after the team commented on realizing the value of the DCs, they were asked "What was the value?" and the responses were "Just for the later stage, and actual physical design." *(22)*, and "It helped in the implementation. How we were going to prompt the user." *(23)*. A third mention related to communicating to system maintainers *(48)*, and the last indicated confidence in the coding stage *(36)*.

Product. Mentions in this category link the use of the DCs to the structure of the resulting system architecture and dialog structure. The DCs were described by the team as "like a sketch of coding" *(4)*, "a beefed up data flow diagram with the user in it" *(7)*, and "a map of the system" *(24)*. The team made an interesting comparative comment:

- ...we never used those data flow diagrams because they were all disjointed.
- You know. This [the DCs] is at least connected and you could see different levels... *(6)*.

Typical references to the <u>*Control Structure*</u> were:

- It's like diagrams of how the system should work. At what point you would intercept the user to get a response. And based on that response what would be the next step. *(5)*
- You kinda see the flow of everything.
- ...And try to get an idea of what information you did have to prompt the user for... *(6)*.
- ...you saw all the levels of depth. You saw all things that you would really have to do and ask for to perform the functions that you proposed. *(31)*.

The DC vocabulary is not intended to be used as a language for modeling *data structure* and architecture. Nevertheless, the following mention indicates that they helped in conceiving data structure as well as the general hierarchical and modular structure of the system:

- [you saw] which information you needed to determine which file you had to access, what calculations needed to be done on the data." *(32)*

Process. References to how using Dialog Charts put <u>*constraints*</u> on the design noted how "You were forced to put all the ideas you had into something concrete" *(17)*, and how the charts helped the team "to keep a very strong control over what was going on." *(35)*. Comments about the direction and <u>*philosophy of design*</u> were made in mention *(30)*, where the DCs were brought up in the context of the functional decomposition of the system until "you hit every possible situation", and also that it is decomposed according to party *(29)*. Recapping mention *(2)* in design evaluation, the team put forth the idea that the specification of the DCs really needed to be an *Iterative process*.

Attitudinal Patterns. The main theme that cut through the mentions in this category is that the DCs were found to be surprisingly valuable. It is interesting to note that the value was not discovered until the charts were used during stages subsequent to the conceptual/logical design. The team mentioned that the DCs allowed them to feel "confident in our code" and made them feel that the tasks of implementation went fast. Specifically, all of the *Learning* mentions point out that the usefulness of the DCs was not apparent to the team until the later stages of the development process, where they were actually used, e.g.,

> - I think we would have concentrated more on getting those right the first time, instead of going back and having to redo them, not knowing the value of them the first time... cause we did, we went back and did them, like twice. *(16)*.

Three more mentions express the view in a similar fashion. It looks as though the team experienced the value of the DCs when they learned that the tool would concretely guide them in building their system.

Redoing Dialog Charts and some record descriptions was credited with positive *Task performance* in the following mention:

> - ...after we had gone back, the physical [design] worked out very well.
> - Very well, see how fast it went though.
> - Yea, but if we hadn't gone back we would have been stuck.
> - I think we would have really trudged through that one, so it paid off.
> - Yea, that's for sure".
> *probe:* and what did you redo again?
> - Um -- the Dialog charts. That was the main one. And some record descriptions. *(44)*.

The team related to the ease in which menu design is derived from dialog charts *(26)*, and also mentioned the DCs in the context of *Task clarity and comprehensibility*. For example:

> - Because, you saw all the levels of depth. You saw all the things that you would really have to do and ask for to perform the functions that you proposed. *(31)*.
> - The main thing is that it helped us to -- see the control. *(33)*.

The team members certainly seemed to derive *Subjective satisfaction* from using the Dialog Charts. It was expressed in the intent to use the Dialog Charts in the future, in their happy surprise at their usefulness, and in the perception of the DCs as valuable.

The idea that the DCs proved to be useful surprised and pleased the team, and they mentioned it four times. One mention is interesting in particular:

- probably the best way to show the contrast is that in the beginning, like when you first starting programming, they made you do flow charts. and you were supposed to do a flow charts before you programmed, and most people programmed and then drew the flow charts afterwards....
- So, I mean, this was totally the opposite.
- ...that's why it's so surprising. For once, we actually used it further on. *(41)*.

They were also surprised that the Dialog Charts functioned as a mapping tool for system structure: "And it really is a true map, which is -- surprising." *(24)*. The *value and importance* of the dialog charts were mentioned three times, twice in connection with the learning process. For example:

- But I think, when we went to the next step, we realized how valuable they were.
- Right...
- And then we redid them. *(17)*.

4.3. Relationships Among Categories

Generally speaking, a relationship is some co-occurance of categories within a mention. As indicated earlier, mentions were categorized with the minimal number of categories, but in some cases more than one category adequately keyed the mention. Figure 4-3 summarizes the co-occurance of categories in coded mentions. In the following paragraphs we briefly comment on some interesting double-coded mentions in the current set of data.

	purpose	stage	product	process	attitude
purpose	25,27 28	22,23 48	5,33 3	20,29 30,30	10,17 26,38
stage	22,23 48	0	0	0	36
product	5,33 3	0	0	14	24,31 49
process	20,29 30,30	0	14	0	35
attitude	10,17 26,38	36	24,31 49	35	16,18 21,44

Figure 4-3: Tally of Multiple-Coded Mentions, by Category

Purpose of using the DCs linked to **Stage** with respect to communicating information among stages of design. In particular, one mention indicates that the DCs "helped in the implementation" *(23)*. Another mention linked **Product** to **Purpose** in reference to the results of designing the control structure: "It's like a diagram of how the system should work" *(5)*. Yet another evidenced a user orientation while designing the queries: "...you have to know the kind of user you're dealing with and formulate those queries accordingly." *(3)*.

Co-mentions of **purpose** with **attitudinal patterns** occurred 4 times, which interestingly centered on communication. Two such multiple-coded mentions indicate surprise because the DCs were helpful or useful in later stages of system implementation *(10)* and *(38)*. One used the term "valuable" about the role of DCs in "so many" following steps. A fourth mention related DCs to the ease of menu design *(26)*.

Product linked to **process** in a mention that expressed the constraint that the DCs forced them to put their ideas into "something concrete" *(14)*. It also linked to **attitudinal patterns** in three mentions. In *(24)*, the team expressed surprise by the idea that the DCs are a "true map" of the system, and *(31)* relates similarly to clarity of the structure and functions. User orientation in designing the product is expressed in *(49)*, along with the intention to use DCs in the future: "No doubt about that." *(49)*. One **Process** mention linked with **attitudinal patterns**. The DCs "helped us to see the control" as well as "to keep a strong control over what was going on" *(35)*. Four **Attitudinal patterns** mentions linked to other aspects in that category. All four are *learning* mentions, three of which are linked to the *value and importance* subcategory, and the fourth mention was linked with *task performance*.

By now the richness of the data gleaned from this single team's experience is apparent. What do all these observations really mean? In the following discussion section we attempt to interpret our findings and relate them to issues currently on the evolving agenda of conceptual dialog modeling.

5. Further Interpretation and Emerging Questions

Sections 2 and 4 above present and examine the DCs as an approach to the conceptual design of dialogs. This section focuses on the more significant findings of the empirical study, and attempts to discuss them in the broader perspective of conceptual modeling of dialogs. Stated differently a question arises as to whether or not a meaningful abstraction of an interaction is specifiable. In the ongoing debate (e.g., in [37]), some argue that a user/system interaction cannot be usefully abstracted, and that any attempt to strip it of application or implementation detail renders such descriptions worthless.

Our study addresses the question as to whether or not conceptual modeling appears to work in practice. Although it relates to a single team only and is necessarily preliminary, the empirical portion of this study does support the case for conceptual design of dialogs. Designers have addressed, in their reference to the DCs, the fundamental attributes of conceptual models and their use.

Conspicuously, a frequent reference was made by the team to the DCs in the context of communicating between the logical and subsequent stages of system building. This idea relates directly to the essential role of conceptual modeling as guiding the design by establishing the conceptual framework within which the dialog is gradually refined and ultimately translated into concrete data and software structures. Furthermore, the general recognition of the value of the DCs was tied to using the tool as a vehicle for learning -- "going back and modifying" -- and evolving a system description and specification. The team's reference to the use of the DCs as a tool for evaluation is also interesting, since evaluation per se was not part of the project, and the evaluation process requires conceptual design that is directly and easily examinable by users. The team also indicated a number of times that the DCs brought in the **end-users** as a focus of the modeling process, making them an un-ignorable part of the design deliberation.

Modeling in general, and conceptual modeling of computer based implementations in particular, are typically "disturbing" in the sense that they neglect concrete details. Dealing with abstraction can easily create dissatisfaction and frustration. In this light designers' emotional responses to the DCs are very relevant and rather interesting -- seventeen mentions reflected various forms of *subjective satisfaction* with the DCs. The team, members of which had taken part in a number of system development efforts before, expected the DCs to be "ritualistic" like other conceptual design tools. In fact, they expressed surprise that the DCs were actually advantageous and valuable, and that they actually used them during later stages of design.

It is interesting that another frequent comment of the team was with respect to the relationship between the implementation and conceptual designs. As an example, it occurred to the team that the DCs capture the essence of the menu in a convenient fashion. This observation seems to support the idea of developing dialogs through the separate specification of logical, implementation and physical dialog models, a process which has become a cornerstone of system development in computerized environments. The same empirical evidence could be equally construed as supporting the role of conceptual modeling in the dialog framework as the guidance for the actual syntactical realization of the interface.

Given that there is no well-defined, validated theory to guide the evaluation of methodologies and tool vocabularies for designing conceptual dialog models, it is very difficult to assess the effectiveness of our research approach. What could be safely stated is that the approach in this paper to the examination of the DCs' usefulness is comprehensive. However rudimentary in scope, the categorizations put forth in this study appear to encompass the nature of the requirements with respect to a conceptual design tool vocabulary. The methodology for enriching the categorization scheme also appears to adequately capture the descriptions of the process provided by designers.

As befits a qualitative inquiry, one of the significant results of the study is the identification of further research questions. A primary concern is the relationship between *usefulness* and *usability* of a conceptual design tool in general and the DCs in particular. A study of a single situation addresses usability in a limited fashion. Not all situations are amenable to the DCs, so the essence of the question is in ascertaining the limits of the tool's applicability, e.g., what type of design situations are easily expressed by the DCs and which range of applications calls for DC use. There is a need for a rigorous assessment of the relationship between the variety of tasks and contexts in which tools are used and the perceived usefulness of tools. Such an examination will allow the prediction of a tool's behavior in a particular design environment, and also allow the designer to select appropriate design situations for using the tool. The empirical part of this study is currently being repeated with more teams [4]. Ultimately, it is going to address the concepts of usefulness and usability more directly. Upon analysis of more and varied cases it will become clearer in which situations the DCs are perceived more valuable.

6. Conclusion

The broad interpretation of the contemporary dialog framework presents a rather compelling argument for the importance of the dialog structure as the intermediate entity, one that bridges between the two other components, and which encapsulates the linkages between the presentation and the application. The Dialog Charts yield a high-level dialog description that is abstract enough to be useful for more than one implementation technique or strategy. The DCs also combine two types of decompositions in the same hierarchy, namely a functional decomposition, which is a common design practice, and a decomposition of parties, which is a distinct dialog modeling requirement. They model the functional requirements of the system, capture the sequencing and control of the interaction, while clearly differentiating between user gestures (i.e., the inputs) and system responses (i.e., the outputs). The DCs were described with high degree of satisfaction, they facilitated the learning of the system context, and provide a vehicle for communicating design information throughout the process of system development.

Although the target tool in this study is the Dialog Charts, the research is an in-depth study of the dialog design process. How do people go about dialog design? What are the requirements for designing dialog structure and control processor components? Ultimately these insights will form the basis for a set of assessment criteria to guide the development and evaluation of dialog design methodologies, and the development of sounder and more robust human/computer interaction.

References

[1] Ariav, G. and Ginzberg, M.
DSS Design: A Systemic View of Decision Support.
Communications of the ACM 28:1045--1052, 1985.

[2] Benbasat, I. and Wand, Y.
A structured approach to designing human-computer dialogues.
International Journal of Man-Machine Studies 21:105--126, 1984.

[3] Britton, K.H., Parker, R.A., and Parnas, D.L.
A procedure for designing abstract interfaces for device interface modules.
Proc. 5th Int. Conference Software Engineering , 1981.

[4] Calloway, L.
An Approach for Assessing Tools for Designing Dialog Structures: A Study of the Dialog Charts.
PhD thesis, New York University, 1989.

[5] Campbell, D.T. and Stanley, J.C.
Experimental and Quasi-experimental designs for research.
Rand McNally, Chicago, 1963.

[6] P. P.-S. Chen.
The Entity-Relationship Model -- Toward a unified View of Data.
ACM TODS 1, No. 1:9-36, March, 1976.

[7] Cheriton, D.R.
Man-Machine Interface Design for Timesharing Systems.
In *Proceedings: Annual Conference of the Association for Computing Machinery*, pages 362-366. Houston, Texas, October 20-22, 1976.

[8] Date, C.J.
An Introduction to Database Systems, Volume 1, 4ed.
Addison-Wesley, Reading, Massachusetts, 1986.

[9] Davis, F.D. and Olson, J.R.
Integrating User Motivation and Task Performance Theories of Information Systems Design.
Technical Report, The University of Michegan, April, 1986.

[10] Dunnette, M.D. (editor).
Handbook of Industrial and Organizational Psychology.
Rand McNally, 1976.

[11] Foley, J., Gibbs, C., Kim, W.C., Kovacevic, S.
A Knowledge-Based User Interface Management System.
In *Chi'88 Conference Proceedings: Human Factors in Computing Systems*, pages 67--72. Washington, D.C., May 15-19, 1988.

[12] Gaines, B.R. and Facey, P.V.
Some experience in interactive system development and application.
In *Proceedings of the IEEE Vol63(6)*, pages 894-911. June, 1975.

[13] Gaines, B.R. and Shaw, M.L.G.
From timesharing to the sixth generation: the development of human-computer interaction. Part II.
International Journal of Man-Machine Studies 24:101-123, 1986.

[14] Glaser, B. and Strauss, A.L.
The Discovery of Grounded Theory: Strategies for Qualitative Research.
Aldine, Chicago, 1967.

[15] Green, M.
A Survey of Three Dialogue Models.
ACM Transactions on Graphics 5, No. 3:244-275, 1986.

[16] Hartson,H.R., Johnson, D., and Ehrick, R.W.
A Human-Computer Dialogue Management System.
In *Proceedings of INTERACT '84, First IFIP Conference on Human-Computer Interaction.* september, 1984.

[17] Hayes, P.J., Szekely, P.A., Lerner, R.A.
Design Alternatives for User Interface Management Systems Based on Experience with Cousin.
In *CHI '85 Proceedings.* ACM, April, 1985.

[18] Hix, D. and Hartson, H.R.
An Interactive Environment for Dialogue Development: Its design, use and evaluation-or- Is AIDE useful?
In *CHI '86 Proceedings.* ACM, April, 1986.

[19] Jacob, R.J.K.
Survey and Examples of Specification Techniques for User-Computer Interfaces.
Technical Report NRL Report 8948, Naval Research Laboratory, Washington, D.C., April, 1986.

[20] Jensen, K. and Wirth, N.
Pascal User Manual and Report, Second Edition.
Springer-Verlag, New York, 1978.

[21] King, R. and McLeod, D.
A Database Design Methodology and Tool for Information Systems.
ACM Trans. on Office Information Systems 3(1):2-21, January, 1985.

[22] Krippendorff, K.
Content Analysis: An Introduction to its Methodology.
Sage, Beverly Hills, CA., 1980.

[23] Malhotra, A., Thomas, J.C., Carroll, J.M. and Miller, L.A.
Cognitive Processes in Design.
International Journal of Man-Machine Studies 12 no 2:119-140, February, 1980.

[24] Martin, P.Y. and Truner, B.A.
Grounded Theory and Organizational Research.
The Journal of Applied Behavioral Science 22, No. 2:141--157, 1986.

[25] McGrath, J., Martin, Joanne, Kulka, Richard A.
Judgement Calls in Research.
Sage, Beverly Hills, 1982.

[26] Mehlmann, M.
When People Use computers: An Approach to Developing an Interface.
Prentice Hall, Inc., Englewood Cliffs, N.J., 1981.

[27] Miles, M.B. and Huberman, A.M.
Qualitative Data Analysis, A sourcebook of new methods.
Sage, Beverly Hills, CA., 1984.

[28] Moran, T.P.
The Command Language Grammar: A Representation for the User Interface of Interactive Computer Systems.
International Journal of Man-Machine Studies 15:pages 3--50, 1981.

[29] Morse, A.
Some Principles For the Effective Display of Data.
Computer Graphics 13(2):94-101, August, 1979.

[30] Nickerson, R.S.
Why interactive computer systems are sometimes not used by people who might benefit from them.
International Journal of Man-Machine Studies 15:469-483, 1981.

[31] Norman, D.A.
Design principles for human-computer interfaces.
In *CHI '83 Proceedings*, pages 28-34. ACM, December, 1983.

[32] Olsen, D.R., Jr.
Presentational, Syntactic and Semantic Componenets of Interactive Dialogue Specifications.
User Interface Management Systems.
Springer-Verlag, Germany, 1985, pages 125--136.

[33] Olsen, D.R., Jr.
Whither (or wither) UIMS?
In *CHI '87 Proceedings*, pages 311-314. ACM, April, 1987.

[34] Parnas, D.L.
On the use of transition diagrams in the design of user interface for a interactive computer system.
In *Proceedings of the 24th National ACM Conference*, pages 379-385. ACM, New York, 1969.

[35] Pfaff, G.E., Ed.
User Interface Management Systems.
Sprinter-Verlag: Berlin, 1985.

[36] Reitman, J.O.
Expanded Design procedures for learnable, usable interfaces.
In *CHI '85 Proceedings.* ACM, San Francisco, April, 1983.

[37] Rosenberg, J., Hill, R., Miller, J., Shewmake, D.
UIMSs: Threat or Menace? (Panel).
In *Chi'88 Conference Proceedings: Human Factors in Computing Sys tems*, pages 197. Washington, D.C., May 15-19, 1988.

[38] Shneiderman, B.
Software Psychology: Human factors in computer and information systems.
Little Brown and Co., Boston, MA., 1980.

[39] Spradley, J.P.
The Ethnographic Interview.
Hold, Rinehart and Winston, 1979.

[40] Teorey, T.J., and Fry, J.P.
The logical record access approach to database design.
ACM Computing Surveys 12, 1980.

[41] Wasserman, A.I. and Shewmake, D.T.
Rapid Prototyping of Interactive Information Systems.
Technical Report, Medical Information Science: UC, San Francisco, 1982.

[42] Williges, B.H. and Williges, R.C.
Dialog design considerations for interactive computer systems.
Human Factors Review: 1984.
Human Factors Society, Santa Monica, California, 1984.