DESIGNING CONCEPTUAL MODELS OF DIALOGS: A CASE FOR DIALOG CHARTS

by

Gad Ariav
and
Linda-Jo Calloway

Information Systems Department
Graduate School of Business Administration
New York University
90 Trinity Place
New York, N.Y. 10006

September 1987

Center for Research on Information Systems
Information Systems Area
Graduate School of Business Administration
New York University

Working Paper Series

CRIS #163
GBA #87-89

Abstract

The conceptual design of user interfaces focuses on the specification of the structure of the dialog, independent of any particular implementation approach. While there is common agreement with respect to the importance of this activity, adequate methods and tools to support it are generally unavailable. The Dialog Charts (DCs) presented in this paper address this problem -- they support the conceptual design of dialog control structures. The DCs combine visual modeling (i.e., diagraming) with widely accepted design principles and an explicit model of dialog structures.

As no clear evaluation criteria exist in this evolving area of dialog design, the preliminary assessment of the DCs takes the form of contrasting them with representative alternative design tools based on Augmented Transition Networks or Backus-Naur Form grammars. The DCs overcome some of the problems that seem to limit the usefulness of comparable approaches. An empirical investigation of the usable power of the DCs is currently underway at New York University, and a summary of this research activity concludes the paper.

1. Introduction

Contemporary paradigms for dialog management identify three sub-components: the syntax processor, the control processor and the functional or applications processor (Figure 1-1). The syntax processor ensures the validity of user inputs and delivers outputs to the user. The applications processor invokes the required application modules and delivers and receives information to and from these modules. The control processor keeps a description of the permissible dialogs and enforces the dialog structure. It also keeps track of the sequencing of processes and maintains the contextual information about the interaction. This conceptualization is consistent with a variety of contemporary dialog models (e.g., [23], [3], [20], [9], [1], [10] [21], [7] and [8]).

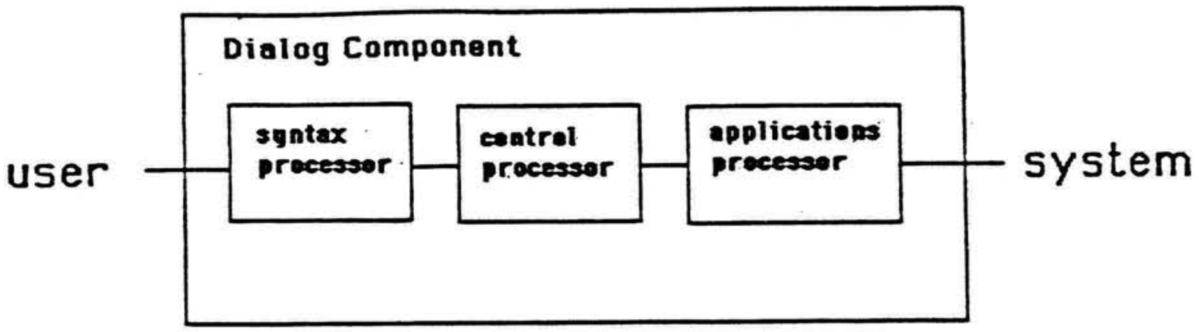


Figure 1-1: The generic structure of the dialog component

One implication of the three-partite model of dialog management is that the design of the dialog structure of a system can be handled independently of the design of the application and the syntax processors. The control structure of the dialog represents, therefore, an abstraction or conceptual model of the dialog. A conceptual dialog model outlines possible sequences of system/user interactions without being bound to a specific implementation method or approach. Such models allow designs to be examined for correctness, consistency and simplicity prior to (expensive) implementation.

Several methods have been suggested for modeling and specifying human/computer interactions including their control structure. In general these methods are oriented towards programmers -- they address the implementation and the physical aspects of dialog design, and therefore only indirectly support the design of conceptual dialog models. Furthermore, these approaches are commonly applied in representing the results of dialog design rather than supporting the design activities themselves. The 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 [22]). The Dialog Charts (Abbreviated henceforth as "DCs") address this problem.

The DCs were conceived primarily to facilitate the design of conceptual dialog models that are structured, easy to manipulate and flexible. They provide a vocabulary of operations, symbols and forms that support the search for alternative solutions in solving dialog design problems. The conviction espoused in this paper is that a useful approach to the design of dialogs' control structure should reflect adequately balanced attention to both **design** and **dialog structures**. The DCs manifest such balanced attention [2].

The DCs are introduced in Section 2, and their use is demonstrated in Section 3. In Section 4 the DCs are contrasted with comparable approaches such as transition networks and modified Backus-Naur Form grammars. Further research concerning the development and evaluation of tools and methodologies for dialog design is outlined in Section 5.

2. Dialog Charts -- Notations

Dialog Charts constitute a tool for solving dialog design problems, and as such they embed an explicit view of both dialogs and design processes. Specifically, the concepts formulated as the framework for the Command Language Grammar [17] 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 [15]. Finally, the types of control flows in the DCs and their diagramatic nature correspond to some key notions of the syntax charts [13]. A more complete discussion of the DCs viz. its underlying "ideologies" is included in [2].

The Dialog Charts were initially developed in 1982 and were used since then in dozens of system development projects where interactive decision support systems and online database systems were designed. The work of selected teams has been studied more carefully as part of an ongoing research on the patterns of use of DCs by actual designers [4]. A brief summary of this research is included in Section 5 below.

Six distinct constructs make up the DC notation. In order to facilitate reference and manipulation of these constructs, they are associated with sets of graphic symbols (see Figure 2-0). Specifically, the constructs are:

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 (indicated by 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.

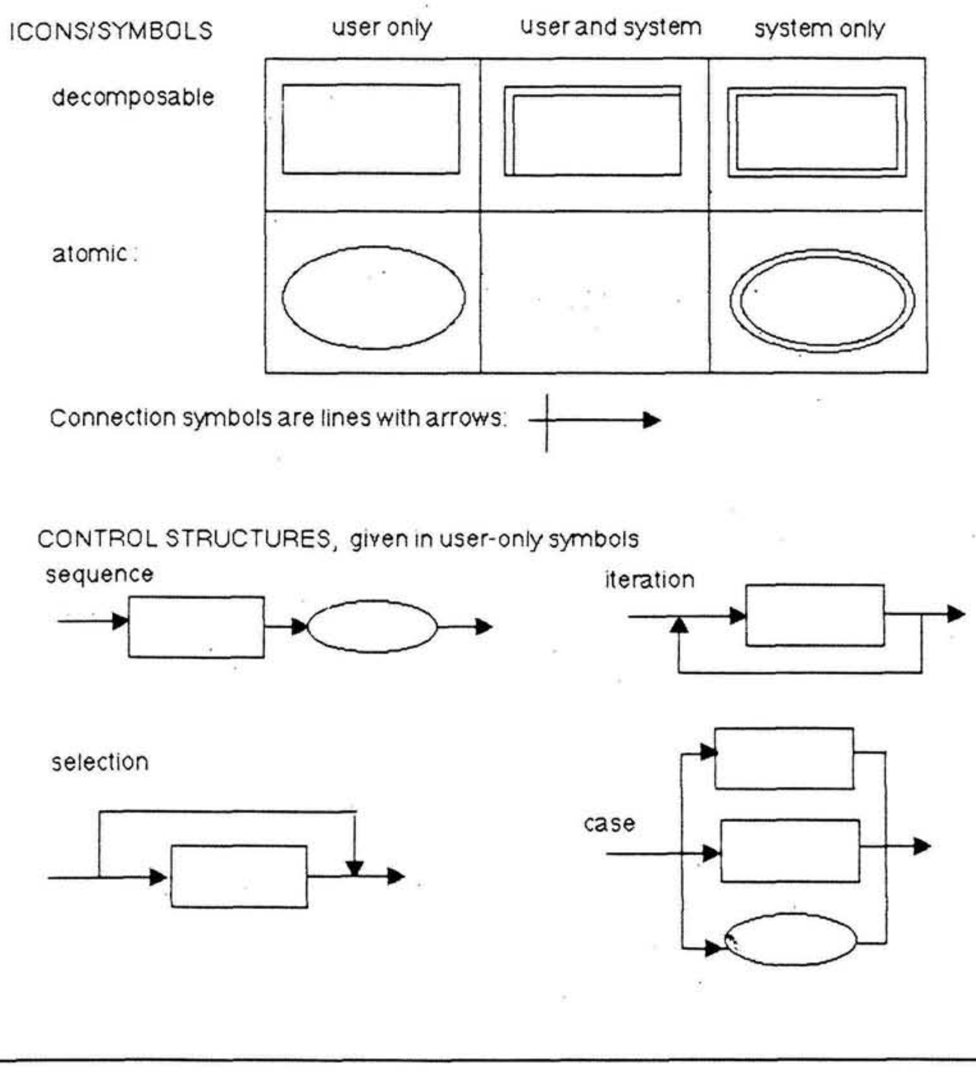


Figure 2-1: Dialog Charts notations and icons

Any box can be further decomposed. It can be 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. A classical issue in design, and especially in conceptual design, is how deeply should the structure be decomposed. There is no clear stopping rule for the elaboration process, although experience suggests that the process should stop when (1) there are no more decomposable elements, (2) further decomposition does not seems to offer new relevant insight.

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. Junctions in the diagrams represent decision points, and are resolved by whoever 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 selects the actual dialog path to be followed.

The specification of error handling procedures within the general structure of a dialog tends to obfuscate designers' and programmers' view of the underlying structure. To avoid this confusion, the DCs support the concept of designing only the permissible dialogs. Only the accepted, proper flows through a dialog are considered. Error handling procedures are added to the DCs as annotations at the appropriate system level. If a procedure applies throughout the system it is stated at the highest level of specification, and if it applies only to a specific junction it is noted at that junction only.

3. Dialog Charts -- Demonstration

The use of the DCs is demonstrated through the conceptual design of a LOGIN command in a Military Message System (Section 6.4 in [11]). In this LOGIN task a user enters into a dialog with a computer in order to establish a session. The goal is to design the interaction in such a way that user/system actions or sub-tasks, are parceled out between the two parties in a manner that meets the specifications of the system.

The LOGIN 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 request 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 [11]). Note that the quote is somewhat ambiguous.

The DCs for that scenario are provided in Figures 3.1 through 3.3, with each figure representing a different level of system elaboration. Figure 3.1 represents the top-most view of the session. It allows the designer to partition clearly the overall flow into well defined concerns.

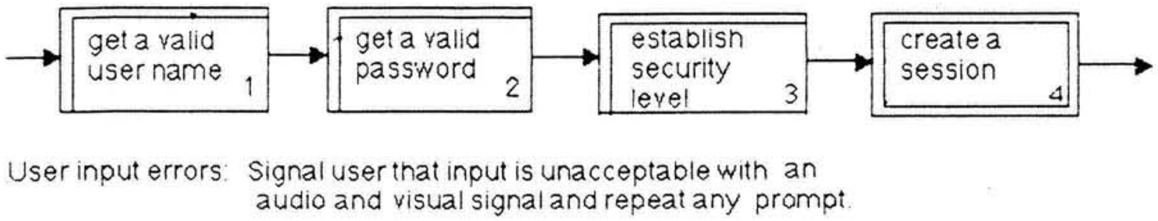

get a valid user name 1
get a valid password 2
establish security level 3
create a session 4
User input errors: Signal user that input is unacceptable with an audio and visual signal and repeat any prompt.

Figure 3-1: MMS LOGIN Session, Top-most level DC

In some cases first level elaboration may be adequate, and the process could stop right there. In order to gain more insight into the LOGIN procedure a further elaboration could be worked out. Figure 3.2 includes two successive levels of elaboration for the box numbered 2 in Figure 3.1, and Figure 3.3 represents a second and 3rd level "explosion" of the box numbered 3 in Figure 3.1.

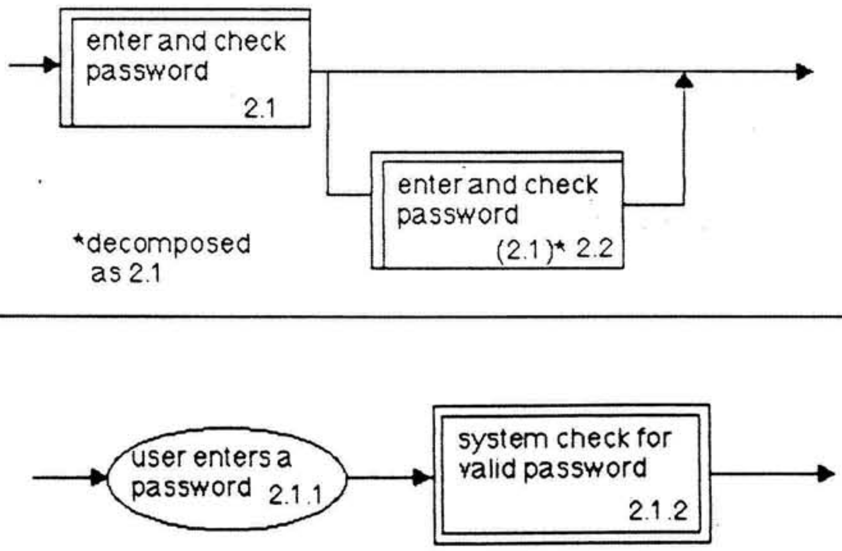

enter and check password 2.1
enter and check password (2.1)* 2.2
*decomposed as 2.1
user enters a password 2.1.1
system check for valid password 2.1.2

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

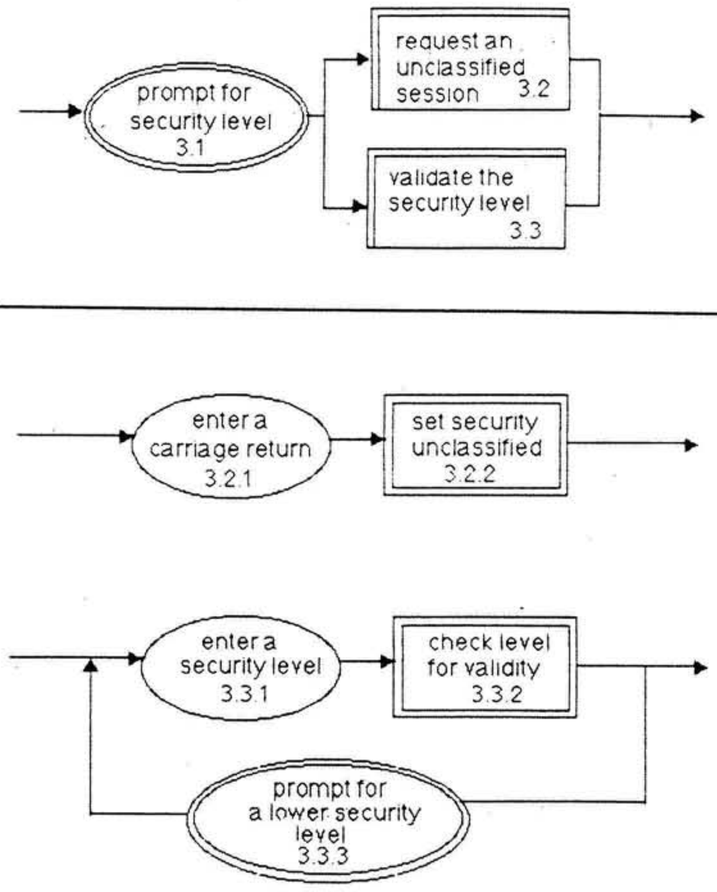

prompt for security level 3.1
request an unclassified session 3.2
validate the security level 3.3
enter a carriage return 3.2.1
set security unclassified 3.2.2
enter a security level 3.3.1
check level for validity 3.3.2
prompt for a lower security level 3.3.3

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

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 enter a Carriage Return to indicate no security clearance, or is allowed to enter a valid security clearance level.

The complete set of DCs for the LOGIN session in any particular design will show which modules and subroutines need to be programmed. The collection of the double boxes and ovals therefore serves as a preliminary input to the detailed design of the applications and the application processor. Furthermore, extracting only the single-lined elements from the charts provides a blueprint for detailed design of the corresponding the syntax processor.

4. DCs in Perspective -- A Preliminary evaluation

One manifestation of the early formative stage which characterizes the area of conceptual modeling of dialogs is the lack of commonly accepted criteria for assessing different modeling approaches. Preliminary evaluation can nevertheless be accomplished through the critical comparison of different approaches, which is the subject of this section.

Generally, there seem to be three typical approaches for designing dialog structures:

- *Analytic methods* employ an abstract and (at least somewhat) formal representation of the interaction along with rules for manipulating the representation. These methods are often based on linguistic models and use a representation such as a modified Backus-Naur Form (BNF). The other typical model is the finite state automata, represented by a modified state transition diagram.
- *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., [5], and [16]).
- *Guidelines sets* are loose collections of principles, policies and rules to be used in "proper" dialog design (e.g., [29], [6], [18], [19], [25] and [26]). A "guideline" advices about the proper conduct for the dialog; for instance, "Control should always remain with the user."

The Dialog Charts belong primarily to the analytic category, and therefore are contrasted here with other analytic methods. The discussion uses a single example (taken from [8]), for which three conceptual models are formulated, using in turn a BNF-based grammar, ATNs and DCs. The task in this example is the "rubber band" drawing of a line on the screen. In this type of drawing, a line is anchored at one point, and extends to the current position of the cursor. It moves with the cursor until a user gesture nails it down and fixes it.

4.1. DCs and BNF-Based Grammars

Backus-Naur Form is a common example of a production rule grammar. Languages are described with this grammar as a set of rules, each specifying a substitution of a composite term by its constituent terms. The chain of substitutions eventually results in a fully specified string in the language [11]. Since dialogs are carried out through preordained expressions ("languages" of sorts), this type of grammar has an obvious appeal in the modeling of dialogs.

Reisner's Action Language Grammar [24] is a typical analytic methodology for dialog modeling. It uses a BNF notation to define a grammar that describes actions taken by the user while interacting with the system. The Multiparty Grammar [27] extends the Action Language Grammar, and allows the designer to explicitly identify human actions and computer actions. The model in Figure 4-1 is a conceptual model of the rubber-band dialog, expressed through a multiparty dialect of BNF.

Control in BNF-based models is at best implicit, and the models do not typically include computer responses. Moreover, as noted by [11], these representations can not explicitly represent control structure. They also describe all possible dialogs that are grammatically valid, however meaningless to the system or user [24].

```
line → button end_point

end_point → move end_point
          | button
```

Context-free grammar for rubber band line example.

```
line → button d1 end_point

end_point → move d2 end_point
          | button d3

d1 →
      { record first point }

d2 →
      { draw line to current position }

d3 →
      { record second point }
```

Rubber band line example with program actions.

Figure 4-1: BNF models for "rubber band" task (From [8])

For comparison, Figure 4-2 includes a DC model for the rubber-band dialog. It only provides a top-level view of the interaction, which seems to fit the level of complexity (or the lack of it) in this specific user/system exchange.

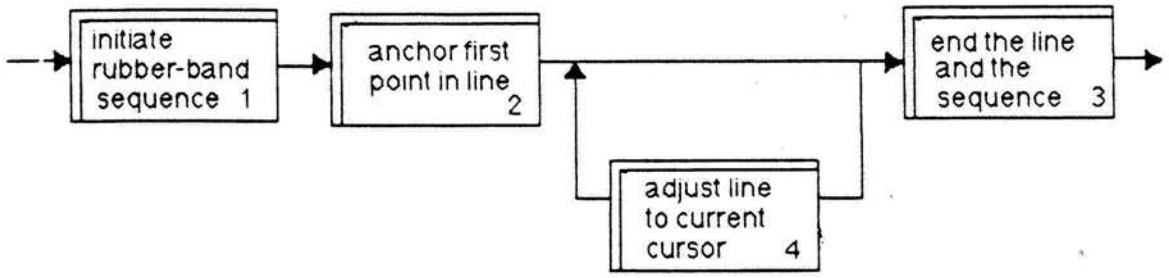

initiate rubber-band sequence 1
anchor first point in line 2
end the line and the sequence 3
adjust line to current cursor 4

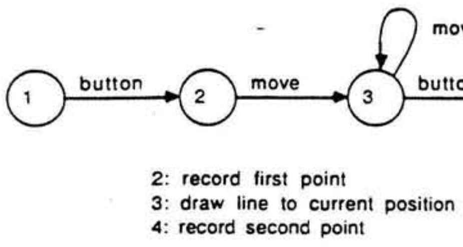
Figure 4-2: A Dialog Charts model of the Rubber band task

The conceptual design, as expressed in Dialog Charts, makes the control structure of the interaction explicit. It is important to notice that the basic flow, structure and connectivity of the design is apparent even *before* any particular interaction syntax is specified. This underlines the functionality of the DCs as a conceptual design tool. Nevertheless, as we saw in the Military Message System example in Section 3, a detailed specification of interaction syntax can easily be derived from the DCs.

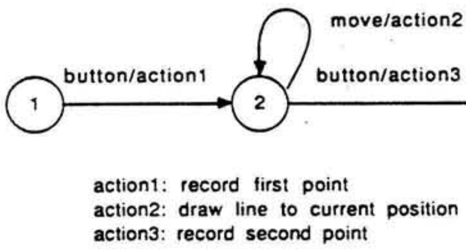
4.2. DCs and ATN-Based Models

State transition diagram are used for describing finite state machines, and they have been used for quite some time to model dialogs [23]. When such a diagram is used to define an interface, the nodes of the network correspond to different states or modes of the Interaction. Arcs linking nodes have one or more input events, output events, or application actions associated with them [10]. This basic form of Transition Networks (TNs for short) has been augmented in a number of different ways with a variety of additional features, forming what is referred to as ATNs. For instance, the nodes may also represent subnetworks, recursion and calls to other nodes [11] [12], *Super states* [5], and *subconversations* [28].

Figure 4-3 represents the rubber-band drawing task as an ATN-based dialog model. Obviously, the DC model is intact, as the task is still the same. The augmentations to the TNs can be viewed as an effort to use them to capture more of the complexities of the various dialog components. The control structure is explicit in an ATN, but it is still obscured by implementation-related details and large numbers of arcs that can form an unconstrained network (as opposed to a structured network). The Dialog Charts, on the other hand, directly address the issue of high level modeling of the control structure, enforce structured network design, and defer the decisions about specific interaction sequences and applications interface issues.



Example transition diagram with program actions.


move/action2
button/action1
button/action3
1
2
3
action1: record first point
action2: draw line to current position
action3: record second point

Example transition diagram with actions on arcs.

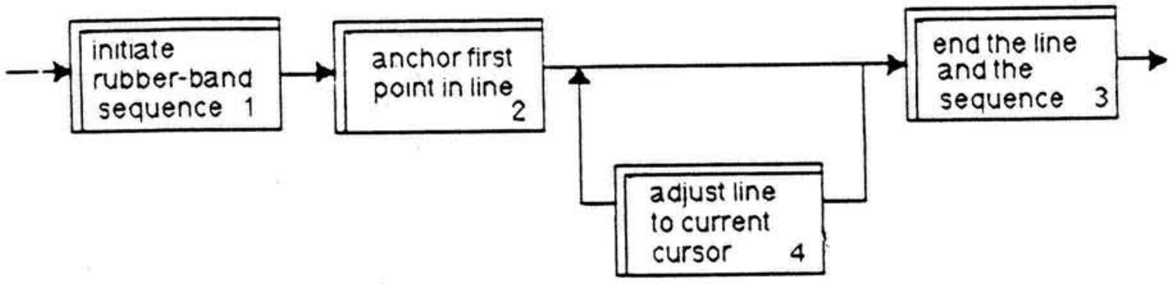
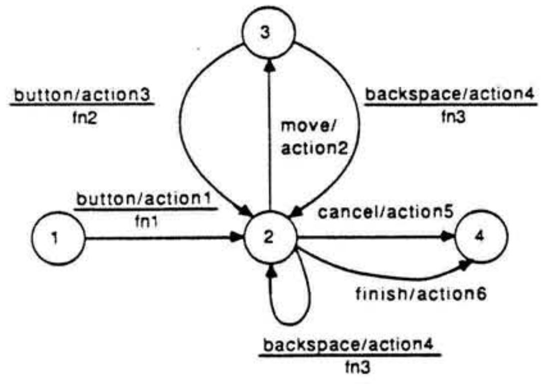

initiate rubber-band sequence 1
anchor first point in line 2
end the line and the sequence 3
adjust line to current cursor 4

Figure 4-3: An ATN model for "rubber band" task (From [8])

Both the problem with obscured control structure and implementation detail become more acute when a more complex task is considered. A slightly more complicated example, extending the line drawing into rubber-band polyline drawing with cancel and backspace options is shown in Figures 4-4 and 4-5. The rubber-band polyline drawing itself is a collection of individual lines created as rubber-band lines and connected end-point to end-point.



```
action1: record first point
action2: draw line to current position
action3: record next point
action4: erase last point
action5: erase polyline
action6: return polyline

fn1: count:=1; return(true);
fn2: count:=count+1; return(true);
fn3: if count = 1 then
        return(false);
     else
        count := count-1;
        return(true);
```

Polyline dialogue with cancel.

Figure 4-4: An ATN model for "Polyline" task (From [8])

Figure 4-4 presents an ATN model for this task. The DCs are more abstract than the TNs and therefore they incorporate the options easily at a higher level abstraction. As has been shown, this structured diagram can be decomposed to specify any particular interaction syntax that is desired.

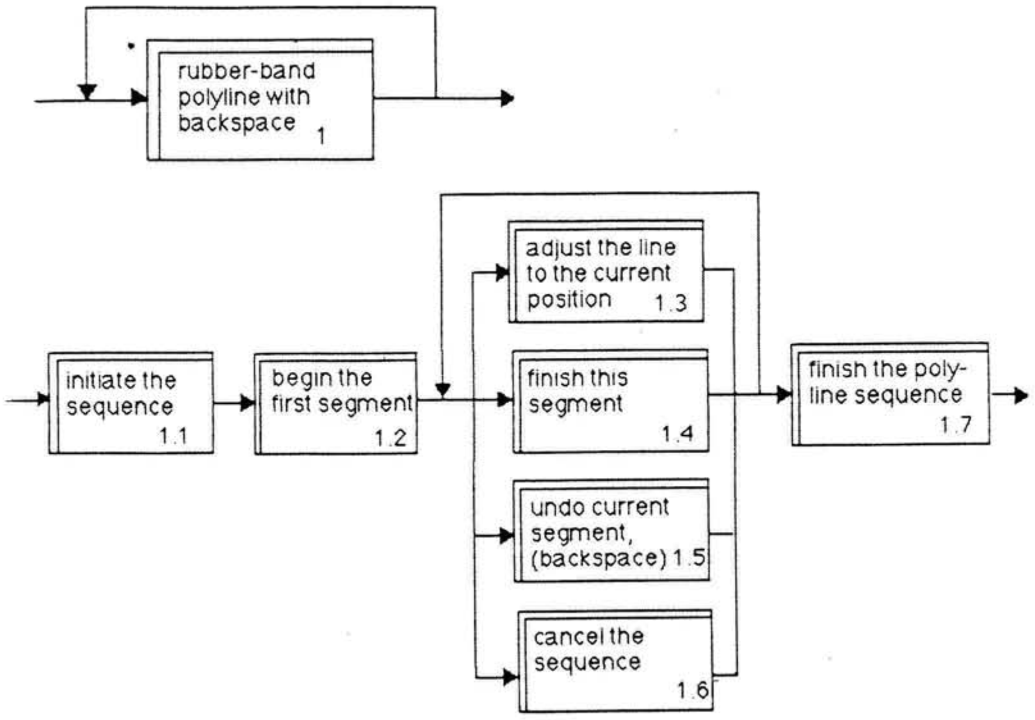

rubber-band polyline with backspace 1
initiate the sequence 1.1
begin the first segment 1.2
adjust the line to the current position 1.3
finish this segment 1.4
undo current segment, (backspace) 1.5
cancel the sequence 1.6
finish the poly-line sequence 1.7

Figure 4-5: A DC model for "polyline" task

5. Conclusion and Further Research

This paper emphasizes the DCs' response to the problems that other comparable tools are collectively exhibiting. In particular, the DCs are yielding high-level design that are 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. The DCs 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) from system responses (i.e., the outputs).

The assessment in this paper is obviously rudimentary and partial. Although Dialog Charts seem promising and were well received by actual users, a more rigorous assessment is required in order to understand their appeal and usefulness. Unfortunately, there is no well-defined, validated theory to guide the evaluation of the various methodologies and tool vocabularies that are used for designing conceptual dialog models. We have therefore opted for a qualitative research methodology in our ongoing research on the use of tools for the conceptual design of dialogs [4].

We have developed a set of research questions that aims to operationalize the notion of usefulness or "usable power" [8]. This set of questions is then used for categorizing the empirical data which were gathered from groups of designers who used the DCs while developing interactive systems. The information was gathered in a non-directed, semistructured interview. The taped interviews are being used in the fashion of archival data, as a source for data-driven content analysis [14].

An informal statement of the basic premise of this research approach is that inferences can be made from the way designers relate to the tool after they have been applying it in actual case. Although the target tool in this study is the Dialog Charts, the research is an in-depth study of the dialog design process. We expect this research to yield the much needed insights into the way people use dialog design tools. Ultimately these insights will form the basis for a set of assessment criteria which can guide the development and evaluation of dialog design methodologies.

References

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

[2] Ariav, G. and Calloway, L.
A Normative Analysis of Approaches to the Conceptual Modeling of Human/Computer Dialogs.
Technical Report, New York University, N.Y., September, 1987.
In preparation.

[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, 1987.
(In progress).

[5] 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.

[6] 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.

[7] Gaines, Brian R. and Shaw, Mildred 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] Jacob, R.J.K.
A Specification Language for Direct-Manipulation User Interface s.
ACM Transactions on Graphics 5, No. 4:283-317, 1986.

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

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

[15] 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.

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

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

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

[19] 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.

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

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

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

[23] 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.

[24] Reisner, P.
Formal Grammar and Human Factors Design of an Interactive Graphics System.
IEEE transactions on Software Engineering SE-7, No. 2:229-240, March, 1981.

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

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

[27] Shneiderman, B.
Multiparty Grammars and Related Features for Defining Interactive Systems.
IEEE transactions on Systems: Man and Cybernetics smc-12, no 2:148-154, March-April, 1982.

[28] Wasserman, A.J., Pircher, P.A., Shewmake, D.T., and Kersten, L.
Developing Interactive Information Systems with the User Software Engineering Methodology.
IEEE Transactions on Software Engieering se-12 No. 2, February, 1986.

[29] 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.

Table of Contents

List of Figures