Using the Dialog Charts Grammar to Complement the Unified Modeling Language to Create Working Prototypes

Linda Jo Calloway
The present paper was presented at the Sixth Annual Symposium on Research in System Analysis and Design held by the Association for Information Systems, Special Interest Group on Systems Analysis and Design (AIS SIGSAND), on May 12-13, 2007, at the University of Tulsa in Tulsa, Oklahoma.

It was originally published in the Symposium's Proceedings on pages 71 - 76.

*Linda Jo Calloway* is Professor of Information Systems, based in Manhattan at Pace Plaza. Dr. Calloway holds a Masters and a Doctorate in Information Systems from the Stern School of Business at New York University. Before returning to university life, Dr. Calloway managed the development of software for real-time telecommunication systems at ITT World Communication and the software for the Apollo launch telemetry data system at Cape Canaveral. She has worked as a researcher and educator in China, Ireland, and the Netherlands.
Using the Dialog Charts Grammar to Complement the Unified Modeling Language to Create Working Prototypes

Linda Jo Calloway
Pace University
lcalloway@pace.edu

ABSTRACT

The conceptual grammar called The Dialog Charts (DCs) has recently been used to complement the Unified Modeling Language (UML) to facilitate the implementation of working system prototypes. The DCs are used to design the decision and control structures of a system, and therefore easily translate into the interactive structures of a prototype system. The DCs appear to be ontologically well defined conceptual models that reflect commonly accepted rules of good decomposition. Masters students in Information Systems created designs for systems from various different real-world situations using the Unified Modeling Language, Entity Relationship modeling and Object Diagrams along with the Dialog Charts (DCs). Anecdotal evidence shows that the dialog charts clarify the actual interaction and decision structure of the system and maintain contextual system information for UML models including use case, collaboration and sequence diagrams.

The students created context diagrams of their systems, UML use case diagrams, fully attributed Entity Relationship models, use-case narratives and/or structured English descriptions of several use cases, use case trigger and outcome matrices, Object Diagrams and the Dialog Charts. The Dialog Charts complement these other modeling tools by providing a simple grammar that describes the system interactivity by designing the decision and control structures that associates the system activities.

This paper recaps the Dialog Charts grammar and begins to evaluate some of its characteristics as a conceptual modeling language according to requirements identified by Wand and Weber (1990, 2002), Paulson and Wand (1992), Gemino and Wand (2003) and Burton-Jones and Meso (2007). The paper then presents an example of how one student used the Dialog Charts. Some conclusions follow with suggestions for future research.

THE DIAGLOG CHARTS

The Dialog Charts (DCs) are a modeling grammar that represent a design of the control and decision structures that connect the activities of the system from the actors’ viewpoints. They also incorporate the ability to decomposing activities performed by both actors and system components together, maintaining a more abstract contextual view of the system.

The DCs are used as an organizing strategy to capture system interactivity at a higher or more abstract level than UML Collaboration Diagrams or Sequence Diagrams. The individual activities within a Dialog Chart may be interpreted by using a Sequence Diagram (to depict the interaction of messages between objects in time sequence) or by a Collaboration Diagram (to depict the flows of messages between objects in message sequence). These two diagrams are not designed for decomposition, nesting or control and so have been used when it isn’t useful to further decompose the Dialog Chart.

The DCs are grammatical as defined by Wand and Gemino (2003). They create representations of the interaction domain of systems using a “set constructs (typically with graphical representations) and rules for combining the constructs into meaningful statements about the domain” (p.80). The DCs also conform to the Good Decomposition
Proposition set out in Wand and Weber's early work on Ontology and automated decomposition (1990, 1992). Specifically, in the DCs every induced event in a subsystem is either (1) External to the subsystem in which it occurs, which is an actor-led transition or (2) Governed by a stability law defined on the state variables of the subsystem, which is a system-led transition. The DCs accommodate Wand and Weber's early premises for decomposable models: 1. the models represent a world made up of things that may be simple or complex. 2. They accommodate two kinds of states, stable states that require interaction from the outside world to transition, and unstable states, that necessitate a transition to a stable state. 3. All things in an unstable state always reach a stable state. 4. The analyst may view the system and its subsystems as being described by collections of state variables (Wand and Weber, 1992).

Serendipitously, the DCs depend on maintaining and defining the control structure of a dialog independent of the implementation approach, and they are easily used to associate use cases with actors or other use cases within an overall decision structure of the system. The dialog charts organize actor/system activities that have not yet been differentiated or decomposed into an abstract decision structure of the system. This abstract structuring can subsequently be decomposed into particular sets of activities, task by task when required. The DCs define a set of permissible dialogs within the structure, keeping track of the sequencing of activities and maintaining the contextual information about the interaction. They allow the designer to defer details of design.

THE DIALOG CHART NOTATION

The idea of sweeping up the various possible decompositions into composite, non atomic structures that may be further decomposed in any order, at any time, to any level of specificity including the actual gestures involved in a particular dialog, appears to be extremely useful as a method of conceiving the system as a set of nested interactions.

The Dialog Chart grammar is simple (see Figures 1 & 2). There are three concepts: (1) an activity may be initiated by an actor, a system component or a combination of both. (2) Any activity may be decomposable or atomic. (3) Activities are combined using program control structures of sequence, selection, iteration or case, such as those described by Jensen and Wirth in the original Pascal User Manual and Report of 1978. The particular symbols that are used are shown below, although other distinct symbols could easily be substituted.

b) A double rectangle is an activity performed by a system component alone.

c) A rectangle with double lines on two sides is an activity performed by both actors and system components.

i) If the double lines are on the top and left, then the actor has made the decision.

ii) If the double lines are on the bottom and right, then the system has made the decision.

2) Decomposition

a) Anything square is decomposable.

b) Anything oval is atomic.

i) A single lined oval is an atomic actor only activity. It won't be further decomposed in this design iteration.

ii) A double lined oval is an atomic system only activity. It won’t be further decomposed in this design iteration.

3) Activities are combined using programming control structures

a) These flows must conform to those commonly associated with structured programming: sequence, selection, iteration and case.

b) A line with an arrow shows the direction of flow.

The decision required to transit a control flow can be written on top of the line if desired. Structured flows aid in robust decomposition and modular dialog design. Similar discussions motivated including these structures in other early dialog modeling methods (e.g., Benbasat and Wand 1984).

![Figure 1. The Dialog Chart Symbols](image-url)
Figure 2. Control Structures in the Dialog Charts (Using Actor only symbols)

AN EXAMPLE USING THE DIAGLOG CHARTS WITH UML

The following example is drawn from the analysis and design of a subscriber information system created for a licensed telecommunication company. It offers its clients a package of TV programs and broadband cable Internet access using a modern digital hybrid optical-coaxial network. The company is planning to offer a new service - IP telephony, and additional information resources are required (Adapted from M.D. 2005).

Figure 3 is the system use case diagram. The Dialog Charts in Figure 4 show two typical approaches towards the design of a prototype. Figure 4.1 shows the approach of having the top level access to the system determined by selecting the subsystem. From Figure 3, notice that there are four subsystems: Subscription, Technical Support, Operations and Supplyment [sic]. The next chart, Figure 4.2, shows the initial system access by the actor. Notice from Figure 3 that the actors include Potential Subscriber, Subscriber, and Time. The actors responsible for the system in the company are grouped under Staff. The first three sets of activities are initiated by the actor as shown by actor-led symbols. The fourth set of activities associated with the actor Time is initiated by the system as shown by the System-led symbol.

Figure 3. Use Case Diagram

Figure 4.1. Access to System by Subsystem
Since one values of the DCs is that they are decomposable and each box may be decomposed separately while leaving the other boxes within a higher level control structure. The example from Figure 4.2 is further decomposed in Figures 5.1 through 5.5.

The Figure 3 use case diagram requires a Subscriber to access use cases from the Subscription and the Technical Support Subsystems. Figure 5 is the Dialog Chart of the activities engaged in by a potential (or new) subscriber and by an existing subscriber.

1. Potential (new) Subscriber: A Potential Subscriber may make inquiries about services or request a subscription to a service.

2. Current Subscriber: A Current Subscriber must Log in, and then may make service inquiries, subscribe to services, change their profile or access technical support.

Figure 4.2. Access to Systems by Actor

Figure 5.3. Current Subscriber Options

Log in for current subscriber.

Figure 5.4. Log in sequence for Current Subscriber

2.2 Subscriptions: The following activities may be accessed from selecting subscriptions

Figure 5.5. Decomposition of Subscriptions

Figure 6 shows the first screen from the prototype. The prototype is fully functional.
It's practically self-evident how the DCs translate into the final prototype.

**Subscriber Services System**

To show relative complexity, the System Context Diagram is shown in Figure 7 and the Object Model is shown in Figure 8.

**DISCUSSION AND CONCLUSION**

The Dialog Charts provide a structured, nested and decomposable context for UML use cases. At any time, a use case interactivity may be diagrammed using other methods such as Collaboration Diagrams or Sequence Diagrams. The Dialog Charts also overtly model interaction activities, and therefore may uncover new interaction objects and use cases.

The Dialog Charts appear to be useful to the designers. Using Gemino and Wand's description of evaluating usefulness, the DCs gave the designers the "ability to represent, communicate, and develop understanding of the domain" (2003, p 79). The congruence between the final Dialog Charts models and their prototypes shows the ability of the grammar to capture the information about this implementation domain and attests to the expressiveness of the DCs (Gemino and Wand 2003).

**FUTURE DIRECTIONS**

As Burton-Jones and Meso note, "Phenomena are not "out there" for analysts to identify. Rather, analysts have to use explicit and/or implicit principles to perceive and conceive real-world phenomena..." (2006 p 39). Although the DCs grammar is simple and seems harmless, the designed...
prototypes reflect certain structuring principles and therefore carry with them any explicit or implicit opportunities and problems that the DCs may create.

It would be useful to investigate the typical ways the designers used the DCs to structure the prototypes. Calloway’s usability analysis of the grammar in 1989 indicated that inexperienced users found the DCs useful as an overarching mechanism to track their work from early analysis through design and prototype development (Calloway, 1989). Calloway’s grounded theory analysis of how these designers used the DCs revealed they were used to communicate among team members and to parcel out development work. These are indications that the DCs may be used to supplement project management.

Some questions require investigation are: What empirical studies might lead to interesting results? Is the structure of the final system clear? Do the diagrams interfere with final designs of objectives? Is it possible or not that the DCs facilitate project management of design efforts?

In the Systems Design and Development course, the DCs facilitated keeping track of activities. As one young designer put it, “They let me know when I was done”.

ACKNOWLEDGMENTS

Thank you to the reviewers and committee members for their excellent support and advice, and a special thank you to the students at Pace University.

REFERENCES


The Ivan G. Seidenberg
School of Computer Science and Information Systems
Pace University
Technical Report Series

EDITORIAL BOARD

Editor:
Allen Stix, Computer Science, Pace--Westchester

Associate Editors:
Constance A. Knapp, Information Systems, Pace--Westchester
Susan M. Merritt, Dean, The Ivan G. Seidenberg School of Computer Science and Information Systems --Pace

Members:
Howard S. Blum, Computer Science, Pace--New York
Mary F. Courtney, Computer Science, Pace--Westchester
Nicholas J. De Lillo, Mathematics and Computer Science, Manhattan College
Daniel Farkas, Information Systems, Pace--Westchester
Fred Grossman, Information Systems; Doctor of Professional Studies, Pace--New York and White Plains
Fran Goertzel Gustavson, Information Systems, Pace--Westchester
Joseph F. Malerba, Computer Science, Pace--Westchester
John S. Mallozzi, Computer Information Sciences, Iona College
John C. Molluzzo, Information Systems, Pace--New York
Pauline Mosley, Technology Systems, Pace--New York
Narayan S. Murthy, Computer Science, Pace--New York
Catherine Ricardo, Computer Information Sciences, Iona College
Judith E. Sullivan, CSIS Alumna, MS in CS from Pace--Westchester
Sylvester Tuohy, Computer Science, Pace--Westchester

The Ivan G. Seidenberg School of Computer Science and Information Systems, through the Technical Report Series, provides members of the community an opportunity to disseminate the results of their research by publishing monographs, working papers, and tutorials. Technical Reports is a place where scholarly striving is respected.

All preprints and recent reprints are requested and accepted. New manuscripts are read by two members of the editorial board; the editor decides upon publication. Authors, please note that production is Xerographic from your submission. Statements of policy and mission may be found in issues #29 (April 1990) and #34 (September 1990).

Please direct submissions as well as requests for single copies to:

Allen Stix
The Ivan G. Seidenberg School of Computer Science and Information Systems
Goldstein Academic Center
Pace University
861 Bedford Road
Pleasantville, NY 10570-2799