DIALOG ANALYSIS IN THE CARNEGIE MELLON COMMUNICATOR

Paul C. Constantinides, Alexander I. Rudnicky

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
pcc@cs.cmu.edu
http://www.speech.cs.cmu.edu

ABSTRACT

In this paper, we present a formative evaluation procedure that we have applied to the Communicator dialog system. In the system improvement process, we have recognized the need to identify interaction failures through passive observation of system use. By systematizing the process of dialog evaluation, we hope to gain a mechanism for effectively communicating descriptions of interaction failures, specifically for use in system improvement. Additionally, we argue that this process can be taught to and executed by an evaluator external to the system development process, with the same proficiency as someone intimately familiar with the mechanics of the system components.

Keywords: formative evaluation, assessment procedure

1. INTRODUCTION

Conversational dialog systems can connect human users with automatic systems in speech based interactions aimed towards the completion of a task. The Carnegie Mellon Communicator System supports speech-only interactions in the travel-planning domain. The user's goal, in an interactive session, is to create a travel itinerary that can consist of multiple flight reservations, hotel reservations, and car rentals. The complexity of this type of planning task is commensurate with the complexity of the itinerary created during the session, and with the disparity between the user's constraints and real world availability.

Communicator, as with any dialog system that supports a planning activity, can involve extended interactions with the user. This type of session can arise from negotiation of solutions to user constraints that may not map directly to the system's view. Interactive sessions with the Communicator can contain interaction failures. This situation can arise when the system's behavior is not consistent with the user's expectations, and could be caused by a variety of factors. A failure could be caused by a bug in a particular component in the system, or could be caused by unanticipated behavior of the user. Interaction failures can hamper progress in the dialog session. At best, these faults leave the user disoriented or dissatisfied with the performance and quality of the interaction; at worst, they can prevent the user from completing the task at hand. Identifying the causes of failure has enabled the system to handle increasingly more difficult scenarios, and to broaden data collection by becoming accessible to more users.

The work discussed in this paper motivates a process for assessing dialog systems. Our aim is to systematize the process of dialog analysis with the following specific goals:

- discover all instances of interaction failure in a particular session
- describe a procedure for annotating these failures that encourages efficient documentation of error
- specify the procedure so that it can be taught to an assessor who is not intimately familiar with the implementation of the system

The procedure we will describe has been used for assessing the Communicator system, and therefore is tailored towards this system, and the travel-planning task. We believe this process can also be applied in systematic analysis procedures for other systems, specifically as part of a comprehensive evaluation process such as the one described in [6]. The technique presented in this paper is particularly well suited for systems that are publicly available, or receive extensive use, as it relies on analysis of use data.

1.1 Communicator System Overview

Automatic dialog systems consist of a variety of components, where each supports a particular type of processing within the system. [8] details the structure of the Carnegie Mellon Communicator system. In brief, the system consists of the Sphinx decoder, Phoenix parser, a schema-based dialog manager (described in [1]), natural language generation and speech synthesis components, along with additional domain-specific knowledge agents. These domain agents include an airline information agent, a component for reasoning over date and time specifications, and an agent for retrieving a user's preferences. Particular modules or knowledge sources within the system are subjected to regression or performance testing during the development cycle. Examples include the grammar, date time component, airline agent, and decoder with language models and acoustic models. Testing helps gauge the performance of these components independently, and identifies problems within them.

The architecture of the Communicator system is organized to support information flow between the user and the dialog manager. In mixed initiative interaction, these primary participants in the conversation can each initiate a topic for discussion towards completing the task, according to their respective strategies. [1] and [8] describe the dialog management approach that grounds the system strategy.

Other considerations in Communicator system design have been focused on making certain that the user's knowledge of the concurrent activity is updated and consistent with what is actually happening. More concretely, the generation components are structured to inform the user of the system status, and to keep the user aware of what is happening in the session.

2. SESSION ANALYSIS PROCEDURE

2.1 Background

Here, we specify a process for recognizing and recording interaction failures in dialog sessions from the Communicator system. This assessment process is grounded by work that has been done for analysis of safety-critical systems. Safety cases and hazard analysis (described in [9]) motivate intentional and explicit reasoning to justify system structure and operability. These well-known techniques are widely used to document and certify safety in hazardous systems. The dependability case technique for system analysis and assessment (presented in [4]) extends and develops these procedures into a process geared towards applications beyond safety and mission critical systems. This technique motivates reasoning and methodology for assessing dependability within a system, encouraging the use of cognitive tools in this process. The dialog assessment procedure given here uses this process as a springboard, using a similar motivation and approach for detecting and categorizing interaction failures. The experiments described in [4] demonstrate the effectiveness of the Ishikawa, or fishbone diagram, that we have adopted for this procedure. This diagram, described in [3], was developed for quality control applications, to visually organize causes-and-effects in a particular system.

2.2 Process Overview

For this process, we will delineate which information sources can be used for analysis, describe some tools that can be used in the analysis, and describe the procedure for annotating failures. The procedure is described as follows:

1. select a dialog and organize information sources for that dialog
2. scan the dialog turn by turn observing user and system actions, looking for failure
3. at each failure, determine a characterization for the failure, noting the conditions under which it occurred

For each dialog, the assessor has a set of information sources over which to perform the analysis. In Communicator, dialog sessions are logged with the audio from each of the user's utterances, and a log file containing the decoding of the user's utterances, the semantic parses of each of these utterances, and a trace of information passed between domain agents. This trace includes the system generated speech act, the associated text output, and requested and retrieved flight information. Along with a transcription of each utterance, this comprises the information available to the assessor.

Using this data, the assessor is tasked with determining where, in a given dialog, an error occurred. Interaction failures and system errors can be characterized by some of their results on the dialog:

- the itinerary generated at the end of the session did not match the user's specifications
- the system did not attend to the user's action
- the session terminated prematurely
- the user expresses frustration or confusion through the course of a dialog
- the system generated inappropriate output to the user

These characteristics are used to determine the points in the dialog where the fault first becomes evident. At this turn (defined as a system utterance and user response pair), the assessor is asked to determine the cause of failure. We present two cognitive aides to assist the assessor in this process. One helps compare the user's goals with the system's goals at that point in the dialog. The other visually organizes failures into categories that will be used to describe the type of error that occurred.

2.3 Turn Analysis

We have developed a code for comparing the user and system goals at each turn in the dialog. Each type of information that the user can specify is given a label (e.g. L1AC corresponds to leg one arrival city, this technique is described in more detail in [2]). The turn is annotated by what information the system requested, and what information was being provided by the user. Noting a failure in the dialog, the assessor uses this tool to help determine where the failure first occurred and to help determine the nature of the failure.

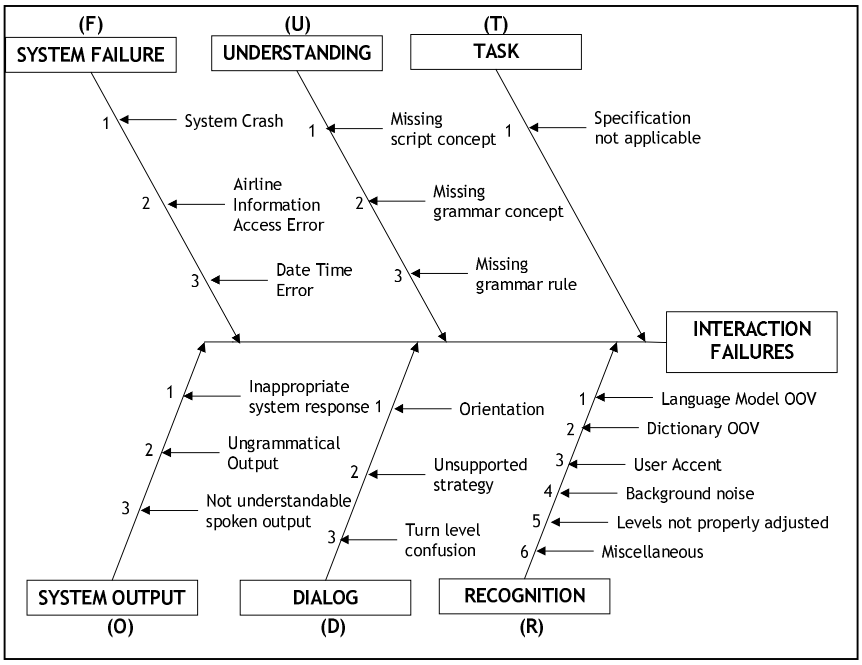

Figure 1: This diagram categorizes some different causes of interaction failures that can occur in Communicator.

The diagnostic stage of analysis involves the use of a visual categorization of sources of system failure, or interaction faults. The fishbone diagram in Figure 1 organizes observed causes for interaction failures into conceptual units. These groupings loosely resemble system structure, and correlate with Communicator system elements. The failure categories labeled System Failure and System Output pertain to failures where some problem prevents the system from giving adequate or accurate feedback to the user, whereas the other categories, however, contain failures caused by mishandling of the user's input. The Recognition category pertains to the decoder and lexicon, while the Dialog and Understanding categories apply to the script and grammar. For each exemplar in this diagram, the assessor is given an example, using dialog features that typify the instance of that exemplar. Additionally, for each exemplar, the user is provided with a question to help determine if this failure is captured by the exemplar. Here is an example of this information, for error cause U2:

Error Label	U2
Error Name	Missing Grammar Concept
Example	User decoded utterance: "what airport am I flying from"; Empty parse.
Refinement question	Is the user specifying something that was not parsed? Are you aware of a phrasing for this specification that would be parsed properly? If not, this is a U2 error.

The annotator is also asked to briefly describe how the conversation ended using the following labels:

B	the session completed, with an itinerary that was not consistent with the user's requirements
U	the user terminated the call prematurely
G	the session completed successfully, with an itinerary consistent with the user's specification
S	the system terminated the call prematurely

The result of this process, for a given dialog session, is a table where each row contains an interaction failure. The row contains the session number, the turn number where the failure occurred, the type of failure (from the fishbone category), and a brief description of the error. Additionally, the session has a row characterizing the result of the session, as described. In circumstances where detected failures cannot be sufficiently described by the tools, the assessors are asked to flag those instances for further evaluation, with the goal of better understanding the nature of these errors.

2.4 Report Generation

Automatic reporting tools are being used to document the detected errors. We are currently using a web-based bug-reporting tool to aid the collection of system use and failure data for the Communicator project. Using these pages, an assessor can submit a bug to the database with the following annotations:

- a brief description of the problem
- a pointer to the session where the problem occurred
- the portion of the log that demonstrates the problem
- the severity of the problem
- the list of modules that are believed to have caused the problem

This tool closes the loop from assessment to development by tightly integrating assessment into system tracking.

4. EVALUATION

We are in the process of performing a preliminary study on the effectiveness of this assessment procedure. We began this study by selecting some 32 dialogs for analysis. These sessions were collected over two days (13 dialogs from the first day and 19 from the second) and are similar in nature. Most of the callers were first time users, who were unfamiliar with the system, and encountered some difficulty in completing their itinerary. A system developer, who has extensively used this procedure, assessed all of these sessions. These assessments were used as a point of comparison for other assessments. Two other annotators were also asked to perform this analysis, in two phases. These individuals have done extensive work transcribing Communicator sessions, and are familiar with the task and goal of the system. They have not been involved in system development.

In the first phase of analysis, the annotators were given a description of the goals of the assessment, namely to determine points of interaction failure, and to characterize them so that they can be documented as bug reports. They were asked to analyze the first 13 dialogs without any background of the procedure described in this paper.

In the second phase, the same annotators were presented with the formal assessment procedure. The process and goals were explained to the annotators, along with examples for all of the concepts being discussed. They were asked to perform assessment on the remaining 19 dialogs using the process. They were not given any feedback on their results from the first phase, before continuing to the second phase.

4.1 Initial Results

In phase one, 66 turns were flagged as containing problems. Collectively, 72 distinct failures were detected, where the distribution of hits and misses between the different assessors was as follows:

	Annotator 1		**Annotator 2**	
Developer	*Found*	*Missed*	*Found*	*Missed*
Found	29	28	43	14
Missed	8		12	

This preliminary experiment tests the performance of a novice using an informal methodology, versus an expert analyzer using the procedure given here. The value of this experiment is seen in the comparison of the resulting reports. The labeled annotations generated using the procedure specifically attribute failure to a particular cause in system operation, promoting effective documentation.

Analysis of the results from the second phase is ongoing. These figures will be made available as they emerge.

5. CONCLUSIONS

Traditionally, user interfaces are created and improved by a dialog designer. This person necessarily has a significant understanding of the mechanics of the different components of this system, and their organization and interaction. The designer must also be proficient in recognizing the needs of a user, and must be able to use this knowledge to affect change in system operation. Here we motivate the role of an assessor, who is not expert in the details of the system structure, but rather is trained in the process of identifying and categorizing system failures. These skills enable this person to effectively and concisely communicate the recognized failures to persons responsible for improving the various components of the system.

The approach presented allows the assessor to lend qualitative analysis to the failures, permitting detection of session characteristics that are not automatically detectable in this system (e.g. detecting user dissatisfaction through changes in tone of voice). Using the tools provided in this procedure, the assessor can concisely characterize and document the failures. Systematically executing this procedure results in a collection of reports documenting the information necessary for system designers and implementers to better understand which aspects of the system, or what components or knowledge structures, need further development. In the case of Communicator, the process has revealed numerous shortcomings that undermine the user's ability to complete a session. Detecting both problems with straightforward solutions, such as coverage expansions or software bugs, and problems that have more nebulous solutions, such as increasing system understandability or keeping the user interested, this procedure is an invaluable source of information for gauging system usability. Within a framework for system and module assessment and improvement, this technique will continue to guide future Communicator development. We hope that this systematized dialog assessment process, that borrows thoughts from quality control and dependability assessment, will be recognized as an important component of dialog system evaluation.

6. FUTURE WORK

The initial results of this trial motivate further investigation. Does this process expose more failures than an informal process? Can a trained assessor substitute for an experienced developer for doing diagnosis? How is efficiency of system improvement affected by the reallocation of diagnostic labor, away from the development circle? These are questions that could be answered with more comprehensive study. A comparison of developer performance (in spotting and characterizing failure) to non-developer performance on the same task would shed light on some of these questions. Moreover, a study on the incidence and efficiency cost of incorrect categorization would also be valuable.

We are also interested in better understanding the role of automation tools in the analysis and communication stages of assessment. Potentially, with further data on interaction failures, the process of scanning a dialog could be assisted by automatic utilities for presenting the log information comprehensively, and for detecting and highlighting potential problems in dialog.

7. ACKNOWLEDGEMENTS

This research was sponsored by the Space and Naval Warfare Systems Center, San Diego, under Grant No. N66001-99-1-8905. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred. We would like to thank Karin Gregory, Christina Bennett, and Jwan Allen for transcribing and analyzing the Communicator sessions. We also thank Maxine Eskenazi for her valuable feedback on this work, and for her contribution of the labeling procedure.

8. REFERENCES

[1] Constantinides, P., Hansma, S., Tchou, C., Rudnicky, A., *A Schema Based Approach to Dialog Control*. Proceedings of the International Conference of Spoken Language Processing: Sydney, Australia, 1998.

[2] Eskenazi, M., Rudnicky, A., Gregory, K., Constantinides, P., Brennan, R., Bennett, C., Allen, J., *Data Collection and Processing in the Carnegie Mellon Communicator*, To Appear in The Proceedings of the Eurospeech Conference: Budapest, Hungary, 1999.

[3] Ishikawa, K., *Guide to Quality Control*, Asian Productivity Organization: Tokyo, 1982.

[4] Maxion, R., Olszewski, R., *Improving Software Robustness with Dependability Cases*. 28th International Symposium on Fault-Tolerant Computing: Munich, Germany, 1998.

[5] Nielsen, J., Heuristic Evaluation. In *Formal Usability Inspections*, Diane D. Cerra (Ed.). John Wiley & Sons, Inc.: New York, 1994. pp. 25-62, Ch. 2.

[6] Polifroni, J., Seneff, S., Glass, J., and Hazen, T.J., *Evaluation Methodology for a Telephone-based Conversational System*. Proceedings of the First International Conference on Language Resources and Evaluation: Granada, Spain, 1998. pp. 42-50

[7] Rudnicky, A.I. The design of spoken language interfaces. In *Applied Speech Technology*, Syrdal, A., Bennett, R. and Greenspan, S., Boca Raton: CRC Press, 1995. pp. 403-428.

[8] Rudnicky, A., Thayer, E., Constantinides, P., Tchou, C., Shern, R., Lenzo, K., Xu, W., Oh, A., *Creating Natural Dialogs in the Carnegie Mellon Communicator System*, To Appear in The Proceedings of the Eurospeech Conference: Budapest, Hungary, 1999.

[9] Wilson, S. P., Kelly, T. P., McDermid, J. A., Safety Case Development: Current Practice, Future Prospects, In *Safety and Reliability of Software Based Systems*, Roger Shaw (Ed.). Springer: London, 1997. pp. 135-156.