A Multimodal Conversational Interface for a Concept Vehicle

Roberto Pieraccini, Krishna Dayanidhi, Jonathan Bloom, Jean-Gui Dahan, Michael Phillips
SpeechWorks International, 55 Broad Street, New York, NY
{roberto,kdayanid,jbloom,dahan,mike}@speechworks.com

Bryan R. Goodman, K. Venkatesh Prasad
Ford Motor Co., Dearborn, MI
{bgoodma2,kprasad}@ford.com

Abstract

This paper describes a prototype of a conversational system that was implemented on the Ford Model U Concept Vehicle and first shown at the 2003 North American International Auto Show in Detroit. The system, including a touch screen and a speech recognizer, is used for controlling several non-critical automobile operations, such as climate, entertainment, navigation, and telephone. The prototype implements a natural language spoken dialog interface integrated with an intuitive graphical user interface, as opposed to the traditional, speech only, command-and-control interfaces deployed in some of the vehicles currently on the market.

1. Introduction

Some cars commercially available today include a speech recognition based system that allows the driver to control non-critical operations, such as controlling the climate of the cabin, operating the entertainment system and making or receiving phone calls. From the user interface point of view, the systems available today are based on a command and control paradigm and limited dialog capabilities [2]. The driver pushes a button on the steering wheel and then is able to issue a single utterance command, such as *climate control temperature down* or *communications dial 555 123 4567*. The difficulty with this type of user interface is that the driver needs to know all the commands, often consisting of a list of more than two hundred [3] sentences, generally provided with the vehicle user manual. Moreover, often the commands are not natural, but follow an artificial language chosen to optimize the recognizer memory and accuracy performance. (e.g. *phone dial memory <number>, navigation voice guidance mute off,* etc.). The designers of these systems have attempted to make this artificial language as consistent as possible across vehicle functions, but this still requires the user to learn enough about the system to make use of all the functions. More intuitive and intelligent interfaces are being studied in several projects [4][5].

This paper describes the design, test, and implementation of a prototype that realizes a multimodal conversational interface to some non-critical vehicle operations. We see this prototype as bringing at least two levels of improvement to the *command-and-control* speech systems found in some current production automobiles. The first improvement is in the system being conversational, thereby reducing the memory burden on the part of the end-user. The second improvement is in the system being augmented by a graphical user interface (GUI) with a haptic (reactive touch screen) interface. The GUI augments the speech interface by providing the user with additional information such as indications on the state of the dialog, suggestions on what to say, and feedback on the recognized utterance. The prototype was demonstrated on the Ford Model U concept vehicle, first shown at the North American International Auto Show, Detroit, January 2003.

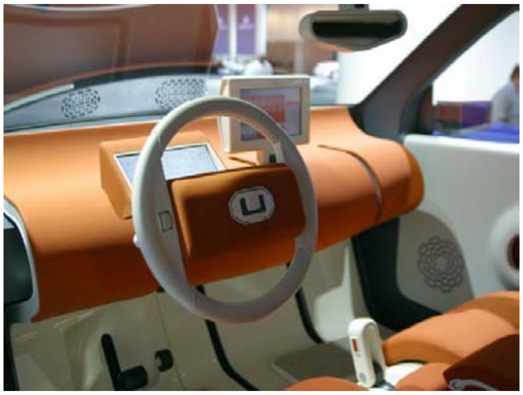

Figure 1: Picture showing the actual vehicle interior with the GUI display mounted on the dashboard.

Figure 1 shows details of the vehicle interior and the touch screen display.

2. User Interface Design

The full functionality of the application is described in Table 1. There are four subsystems that can be controlled by voice: climate, communication, navigation, and entertainment. A fifth subsystem was created to control the appearance of the

CLIMATE: cabin and seat temperature, fan speed, fan direction, recirculation, fresh air, open/close roof, front and back defrost
TELEPHONE: dial by number and name, browse contact list
NAVIGATION:destination entry, points of interest, scroll and zoom map, find current location.
ENTERTAINMENT: play MP3s by category, artist name, playlist name, browse list, change volume
PREFERENCES: stop and activate voice response, change voice (male and female), change display appearance (analog and digital)

Table 1: Functions provided by the system

vehicle gauges, switch the system voice on and off, and change to a different voice. Each subsystem corresponds to a different layout of the GUI, with appropriate widgets designed for controlling the corresponding functionality, as in the example of Figure 2.

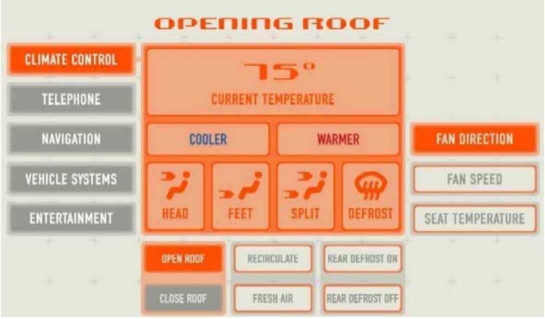

Figure 2: Screenshot of one of the GUI displays

The main principles driving the design of the user interface were reducing the need for the driver to read documentation on how to use the system (i.e. the system should be intuitive) and making the system flexible for advanced users. In addition, the requirements called for a truly multimodal interaction (users would be able to choose between touch screen and voice at any point in time). Finally a basic principle of “not speak until spoken to” was adopted in order to make the system as least intrusive as possible. Given these principles, the UI design adopted the following criteria:

1. The GUI displays the current state of the interaction at any point in time.
2. All the words (and their synonyms) currently visible on the GUI can be spoken.
3. It is possible to switch between subsystems at any point in time by preceding any command with the subsystem’s name. For instance, while being in the climate subsystem, one could make a telephone call by saying: *Telephone, call John Smith*.

We assumed that the spoken interaction with the system could be conducted on one of three different levels.

Directed Dialog: this type of interaction is especially useful for novice users. The system will prompt the user for soliciting new information towards the completion of the task. A new user would start by speaking words visible on the GUI. For instance if the system is currently in the initial idle state, the user would see buttons with words such as *climate control, entertainment, telephone,* etc. If the user says *climate*, the system would switch to the climate control GUI and prompt the user for additional information. The user could proceed by speaking the name on one of the climate buttons, for instance *seat temperature.* Thus the system will prompt for the required additional information, e.g. *Driver, passenger, or both?* After the user’s appropriate response, it would ask: *Warmer or cooler?* In this way the user, with directed help of the system, would be able to explore the space of the possibilities offered by the system and create a mental model of it.

Terse commands: After becoming familiar with the application space, the user can speed up the execution of commands by providing the necessary information in a single sentence, without being repeatedly prompted by the system. For the previous example, the driver may want to get the seat temperature adjusted with a single sentence, and could say something like *Climate driver seat temperature up*. Each interaction that requires more than one piece of information allows for one or more versions of terse commands.

Natural language commands: Most natural language expressions are captured by the grammar at any stage of the dialog. For instance, users can say things like *Turn the seat temperature all the way down*, *I want to call John*, or *I’d like to hear the Rolling Stones.*

For compound commands (both terse and natural language), the system will engage in directed dialog if information is missing for the completion of the task. The system is mixed initiative, in that any command in any of the previous forms can be given by the driver at any point in time, in addition to the possibility of using voice or haptic interaction. For instance the driver can initiate with a natural language utterance, the system will engage in a directed dialog request, which the driver can respond to by clicking a button on the display. This prototype was built to show the full capability of a multimodal system, while safety was not the primary goal. In a real application, the use of the touch screen could be restricted to when the vehicle is at a full stop.

3. Speech Recognition

Speech was collected through an ANDREA DA-310 microphone array that, in the final prototype, was installed in the headliner of the vehicle. The Speech2Go embedded speech recognizer was used with adapted acoustic models, as described in section 8. The static vocabulary consisted of 624 pronunciations, including contact names, MP3’s artist, categories and playists, and a list of 74 cities around Detroit

Dynamic semantic models (DSM) and *conditional confirmation* were introduced in the recognition algorithm in order to improve the speech accuracy and the overall interaction speed.

Application of DSM consists in re-scoring the n-best speech recognition output strings according to their semantic content. The most likely semantics in a particular context gets a higher score. Context is represented by the current state of the dialog. There are a total of 29 different contexts, corresponding to the different dialog nodes that activate a speech recognition collection. For each context a weight is associated to each different sentence meaning (e.g. the sentences *turn the fan off* and *switch the fan off* have the same meaning, thus they get the same weight). Weights are computed during a training phase using a corpus of training utterances.

A simple cache model is also applied by keeping a history of the most recent interactions and modifying the n-best scores according to the observed recognition results.

The final score of the first best speech recognition result is used to decide whether to reject, confirm or accept the utterance, based on a two-threshold schema. While the rejection threshold is kept constant throughout the application in order to minimize the total number of false acceptances and false confirmations, the confirmation threshold changes depending on the meaning of the sentence that has been recognized. The setting of the confirmation threshold attempts at minimizing the number of *costly* mistakes, where the cost of a mistake depends on how far from the current context will the interaction go should we accept that command. For instance, while the user is engaging in a sub-dialog trying to determine a contact to call, the recognition of a sentence like *call her at work* will be considered less costly (being in the same semantic context) than a sentence like *I'd like to listen to classical music.* To that purpose, a higher confirmation threshold is set for sentences which meaning is *out-of-context*.

4. Grammar Development

Each collection state of the dialog (i.e. each state in the dialog that would correspond to a speech recognition action) was provided with a different context free grammar. A total of 5000 utterances, collected during the early stage the system development, were used for improving the grammar coverage. The grammar at each state of the dialog consists of the union of general grammars for each branch of the applications (e.g. climate, navigation, etc), state specific grammar to cover the possible answers to the current prompt (e.g. yes/no grammar for yes/no questions) and general command grammars (e.g. cancel, help). In turn, each specific branch grammar is the union of the natural language and the terse command grammar for that branch. The size of the union of the branch specific grammars consists of about 450 rules.

For the telephone branch a dynamic grammar is automatically generated from the list of contacts allowing for first and last name recognition. The dialog would provide disambiguation in case of homonyms. A spelling grammar is also generated as a fallback in case of recognition failure. Similarly, for the entertainment branch, a grammar is created representing the list of mp3 audio files by category (e.g. Rock, Jazz, Classical, etc.), name of the artist (e.g. The Rolling Stones, Aerosmith, etc.) or by playlist name (e.g. Oldies, The Eighties, etc.)

The navigation module of the system allows for destination entry. In this prototype we restricted the vocabulary to the greater South-East Michigan area for a total of 74 cities, including Detroit and Dearborn. Once the driver specifies the destination city, the system prompts for the address and loads the appropriate address grammar that allows for phrases including number and street names. The number can be spoken as an arbitrary combination of natural numbers. For instance, the number 1345 can be spoken as *one thousand three hundred forty five*, *thirteen forty five, one three four five,* etc. In case of recognition failure (i.e. recognition score too low), the driver is prompted for the number and the street names separately.

5. Speech Output

Speech output consists of a mixture of recorded prompts and high quality concatenative TTS (Speechify Solo) from the same speaker. TTS is used parts of the output prompts, such as numbers, street and contact names. Two sets of prompts (male and female voices) and the corresponding TTS's are used; the driver can choose the voice gender within the *preferences* branch of the application.

6. Multimodal Dialog Management

The architecture of the system, shown in Figure 3, is based on a multimodal version of the ETUDE dialog manager [1][6]. ETUDE is a dialog manager described by a recursive transition network. Nodes of the network correspond to the invocation of functions (action objects); transitions between nodes are ordered and associated to conditions on a frame data structure that includes all the session state variables. Action objects can modify the frame. Both actions and transitions can be arbitrarily complex. An ETUDE network (dialog) qualifies as an action object, thus allowing for a hierarchical definition of the dialog flow (sub-dialogs). Action objects can perform synchronous interaction with system resources (e.g. invoke ASR with a particular grammar, play prompts, change the GUI layout, etc.), perform arbitrary computation, and write information on the frame. Asynchronous events are notified to the dialog manager (e.g. a button click on the GUI) and can cause information to be written on the frame. Events are bound to event handlers that can change the state of the underlying resources. For instance, when a button is clicked while the ASR is active, the associated event handler stops the recognizer, thus causing the dialog manager to perform a transition.

The application is written using the ETUDE API, which allows creating dialogs, nodes, and transitions that describe the dialog flow. Each piece of functionality of Table 1 is represented by a form-filling network topology (Figure 4). This structure, functionally equivalent to the form interpretation algorithm (FIA) implemented by the voiceXML standard [7], allows for mixed-initiative collection of different pieces of information. Mixed initiative is implemented by using ETUDE's global transitions [1]. A global transition to a given node allows the transfer of the dialog control to that node from every node of the current dialog and all the sub-dialogs. For instance, a global transition to the start node of the seat-temperature sub-dialog can be set for the whole application. As a result, if the user requests a change in the

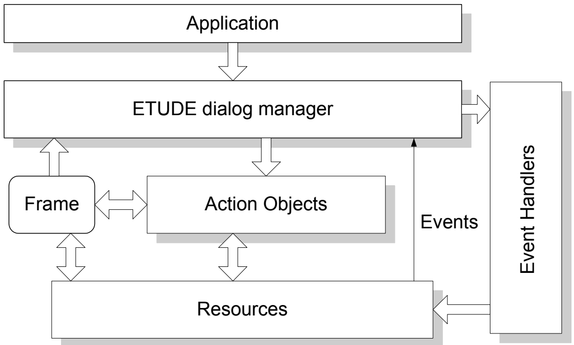

Figure 3: Multi-modal architecture.

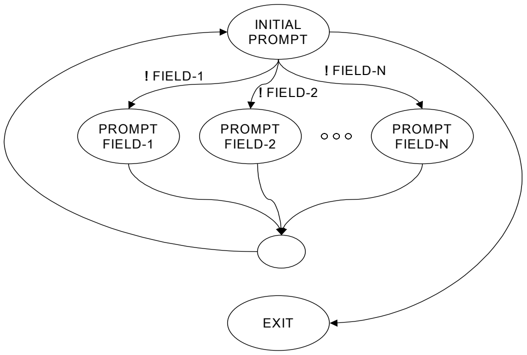

Figure 4: General structure of the form-filling network. Condition on the arcs test whether the associated field is empty.

seat temperature from any node in the dialog, the control is transferred to the seat-temperature sub-dialog.

7. Usability Testing

Usability testing is an integral part of the design, tuning, and development of a user interface. Usability testing is not a substitute for more large scale testing targeted, for instance, to determine the speech recognition accuracy and the task completion rate. Usability testing is focused instead on finding macroscopic problems with the user interface, and suggesting possible solutions; a small number of subjects (between 5 and 10) and a few tests at different stages of the development are generally sufficient. We did perform two usability studies during the development lifecycle of the described prototype. The first one (low fidelity) was conducted after the UI design but before the system was actually developed, using actual screenshots of the GUI and asking the subject to pretend they were interacting with a real GUI and a voice interface. The subjects played a videogame as a distracter task. The second usability test (high fidelity) used an advanced implementation of the prototype. The subjects used a drive simulator videogame while they were asked to perform functions (like reducing or increasing the temperature, playing music or asking for directions). Both tests employed 10 subjects of different age and experience. Results from both tests were used for improving the user interface and the grammar coverage. Among the major findings from the usability standpoint were the difficulty for users to use the push-to-talk procedure, and the need for an idle dialog state (i.e. Main Menu).

8. Experimental Results

The acoustic models used in this system were initially trained from a large general corpus resulting from an actual in-car collection. Successively the acoustic models were tuned (using a MAP algorithm) using 12000 utterances (half male, half female) collected through collection scripts, with 120 speakers each contributing 100 utterances prompted from script. This created a positive bias in our acoustic models to the grammars used in the application. 5000 utterances (70% male) were then collected using an early prototype of the application, which contributed to a better match of microphone and environmental conditions. Overall, we achieved a 25% relative error rate reduction through acoustic model adaptation.

Context	N. Utt.	No DSM	DSM
Climate	1608	89.6	91.5
Telephone	768	94.8	96.4
Navigation	500	96.8	96.8
Entertainment	600	95.3	96.0
Preferences	1112	96.4	96.4
TOTAL	**4588**	**93.7**	**94.7**

Table 2: *Sentence accuracy without and with DSM.*

The 5000 utterances collected with the prototype were also used to improve coverage of the grammars; out-of-grammar rates were then reduced from about 20% to 10%. Tests were conducted on a set of additional 4588 utterances collected with the prototype.

Table 2 shows the effect of dynamic semantic models (DSM) per context averaged over the five different branches of the system and shows a modest but statistically significant improvement of the accuracy.

9. Conclusions

This paper describes the design and development of a conversational, mixed initiative system for controlling non-critical operations on a vehicle. The UI was carefully crafted in order to allow drivers with no experience to use the system without prior training or reading a user manual. The UI allows expert users to speed up the interaction by using compound and natural language sentences. The architecture of the system is based on a multimodal version of a recursive transition network dialog manager (ETUDE). Specific data collection, careful tuning of the grammars, and use of dynamic semantic models allows attaining a satisfactory accuracy.

10. References

[1] Pieraccini, R., Caskey, S., Dayanidhi, K., Carpenter, B., Phillips, M. “ETUDE, a Recursive Dialog Manager with Embedded User Interface Patterns,” *Proc. of ASRU01 – IEEE Workshop*, Italy, Dec. 2001.

[2] Heisterkamp, P., Linguatronic - Product Level Speech System for Mercedes-Benz Cars, *Proc. of HLT 2001*, Kaufmann, San Francisco, 2001.

[3] Jaguar Cars Limited Publication Part No. JJM 18 09 24/22, Coventry, UK, October 2001

[4] Minker, W., Haiber, U., Heisterkamp, Intelligent Dialog Strategy for Accessing Infotainment Applications in Mobile Environments, *Proc of IDS02,* Kloster Irsee, Germany, June 2002.

[5] Bernsen, N. O., Dybkjaer, L., A Multimodal Virtual Co-driver’s Problem with the Driver, *Proc of IDS02,* Kloster Irsee, Germany, June 2002.

[6] Pieraccini, R., Carpenter, B., Woudenberg, E., Caskey, S., Springer, S., Bloom, J., Phillips, M., Multi-modal Spoken Dialog with Wireless Devices, *Proc of IDS02,* Kloster Irsee, Germany, June 2002.

[7] VoiceXML eXtensible Markup Language Version 1.0, http://www.w3.org/TR/voicexml