EXPLOITATION OF CONTEXT INFORMATION FOR NATURAL SPEECH DIALOGUE MANAGEMENT IN CAR ENVIRONMENTS

Gerhard Rigoll and Markus Ablassmeier

Institute for Human-Machine Communication
Technical University of Munich
Arcisstr. 16, 80290 Munich, Germany
phone: +49 89 289-28541

{rigoll | ablassmeier}@ tum.de

ABSTRACT

This contribution focuses on a situation- and user-aware approach of multimodal dialogue management implemented in a framework for the automotive environment. A dedicated dialogue manager for driver's interaction with driver information systems (like infotainment and communication systems) as well as driver assistance systems has been developed and tested. One main focus in the development was the ability to make context-dependent decisions.

The dialogue manager provides flexible and user-centered speech dialogues and support multimodal interfaces, like buttons or turning knobs combined with speech. For the dialogue control, a frame-based approach is used.

The dialogue description is realized in XML which allows for an easy overview over the dialogue structure. Visual outputs are displayed on several screens in the car.

The usability evaluation shows an improvement of effectiveness, a higher joy of use through the possibility of submitting several pieces of information in only one dialogue step with natural speech comparing to a menu-based spoken dialogue. The situation-dependent information assistants reached a high acceptance. The test persons rated the context-based way of frame-based interaction as comfortable and important.

1. MOTIVATION AND INTRODUCTION

In this chapter, we would like to introduce to the field of usability research in the car domain.

1.1 DRIVER TASKS

In modern applications, an enormous increment of various technologies can be realized. Due to the price decline of many electronic devices, their compact size, and their great capability, today's technical systems offer a large spectrum of functionality. Yet, as a direct consequence of the functional complexity, the interaction with such interfaces is getting more and more difficult for the user which often leads to different kinds of operation errors.

Especially in the car domain, often error-prone situations occur regarding human-machine interaction with different in-car applications, as the driver often has a certain mental workload. This basic stress level is due to the execution of so-called *primary* and *secondary tasks*, and may be increased by environmental impacts, like the conversation with a co-driver. Primary Tasks are segmented into navigation, steering, and stabilization. Secondary tasks are operations, like reactions to and dependent on driving demands, but they are not essential to keep the vehicle on track. Tasks not concerning the actual driving itself are categorized as *tertiary tasks*. If the driver interacts, i.e., with a communication and infotainment system in such a stress phase (tertiary task), inattention, distraction, and irritation occur as a consequence of the high workload resulting from a superposition of the tasks mentioned above, which will become manifest in an increased error potential and in erroneous operations of these tertiary systems. [1] and [2] introduce an in-depth classification of driving tasks.

1.2 CONTEXT INFLUENCE

During driving, a large number of influencing variables are effective on the interaction and, as a consequence, on the dialogue between driver and the tertiary systems to be operated. These factors are summarized as context parameters and can be divided into three subgroups: environmental, user, and system context parameters.

Traffic volume, road or weather conditions are continuously subject to change. These factors may strongly influ-

ence the driver's performance and attention, and are referred to as environmental context.
Also the user her- or himself has a strong impact on the dialogue. An expert who is familiar with the system may prefer other display contents and shortened dialogue structures compared to a novice user. Being on a private journey, the driver will have other needs and preferences compared to a daily routine ride (e.g. the trip to get you to your office). Moreover, emotional expressions of the user play an important role for an effective context-adaptation.
The system context can also influence certain dialogue steps. Some systems state conflict with one another. For example, an intelligent navigation system should disable the menu item "start navigation" unless the driver does provide a destination.
In dialogue systems, contextual parameters are applied in three different ways: The form of the dialogue can be changed by adapting the verbosity in dependence of the user state. On the other hand, context can be exploited for making information available to the system. Thus, information does not need to be gathered explicitly from the user. For instance, let the car be in Munich, and the driver just feeds in "Arcis Street," then the city can, by default, be concluded from the context for reasons of plausibility. At last, contextual factors can trigger the system to initialize a dedicated dialogue. An example is an active prompt of the system to evade a traffic jam.

1.3 MULTIMODAL INTERFACES

In many domains multimodal interfaces (MIs) are increasingly applied. If the system has a mono-modal interface, the error aggravates, as the system can no longer be properly used until the cause is found and removed. Besides classical tactile sensors (buttons, turning knobs, etc.), MIs also allow for speech, gestures, or both as interaction paradigms [3]. Compared to mono-modal systems, MIs provide significantly shortened learning periods as well as a highly individual and intuitive form of interaction, since they meet natural human communication habits [3].
If, for any reason, the chosen modality channel is disrupted, the user can be suggested an alternative input form as a fallback in dependence of the given functionality and the situational context. The system could alternatively suggest the driver tactile input, if the car is just waiting at a stop light. Oviatt et al. mention that in dedicated scenarios, it is possible to omit up to 86% of all task-critical errors, if only an alternative input modality is present [3]. When the user provides input in a synergistic and redundant way, mutual ambiguities can be eliminated. To distract the user as less as possible, and, on the other hand, to increase operation comfort, a context sensitive error management is imperative. The goal is to duly identify potential error sources concerning the operation of the applications mentioned above, and to trigger according measures, i.e., warning outputs (a priori-error management). If errors have occurred, they should be solved rapidly, effectively, and transparently for the user to prevent further distraction and error propagation (a posteriori- error management) [4].

1.4 ONLINE SURVEY ON USER'S NEEDS AND DESIRES

This chapter presents an overview of the results of an online survey prior to the development and the evaluation of the dialogue manager presented later on. The goal of the survey was to identify the a-priori attitude of possible users towards utilizing information and assistance systems in cars by speech.
A total of 123 participants, 95 male and 28 female, completed the questionnaire. Basically, the participants have a positive attitude towards voice-controlled applications in cars assumed an absolutely accurately working speech recognition system. Faultless speech recognition would lead to an enormous increase of roadworthiness and comfort while driving. Most suitable are infotainment features, because in this domain, possible recognition errors would not affect the driving behavior of the vehicle. In general, voice control is preferred to tactile operation in almost every case, especially in crucial driving situations. But the preferred modality is context and function dependent. Control processes (e.g. adjusting the volume of the radio) are favored to be done using tactile controllers. If there are passengers on board or music playing the driver prefers tactile und visual devices.
About one third of the interviewees point out that essentially the speech interface has to be able to deal with natural language. Acoustic feedback should be as short as possible. 90% answered, they would like to adjust the verbosity of the spoken feedback.

2. SPOKEN DIALOGUE SYSTEMS

A spoken dialogue system is defined as a computer system which uses spoken language to interact with a user. This interaction aims to solve a certain task. [5]
Dialogue systems include speech recognition, speech synthesis, language understanding, and dialogue management.
According to [5], spoken dialogue systems can be classified into three main types, depending on the methods used to control the dialogue with the user. These are finite state-based, frame-based and agent-based systems.
In the *finite state-based approach*, the dialogue consists of a predefined sequence of states and conditioned transitions between them. In each state, the system prompts for a user input that generally is expected to be a single word. Depending on this input, the evolving dialogue runs on alternative ways through the dialogue graph. In general, this kind of system is only suitable for clearly structured dia-

logues with limited quantity and complexity of user input. On the other hand, technical complexity is rather low.
In *frame-based systems*, the user is asked questions that enable to fill slots in a template in order to perform a task. The dialogue flow is not predetermined, but depends on the user's input and the information the system has to elicit. However, if the user provides more than the requested information, the system can accept this information, and check if any additional item of information is required. The dialogue definition consists of system prompts, together with a condition which has to be true for the prompt to be relevant. Referring to the knowledge sources, frame-based systems require an explicitly defined task model because this is used to determine which question still has to be asked. Frame-based systems provide the possibility to freely decide which and how much pieces of information to enter. This leads to a more natural kind of communication and is essential in case the user doesn't know at the beginning which information is actually needed to succeed.
Agent-based dialogue systems allow complex communication between the system, the user, and the underlying application in order to solve a problem or task. These systems tend to be mixed initiative, which means that the user can take control of the dialogue, introduce new topics, or make contributions that are not constrained by previous system prompts. Concerning these systems, [5] regards communication as an interaction between several agents, each of them capable of reasoning about their own goals and actions. Progressive dialogue context is considered in the dialogue model. In general, there is no given dialogue definition, but the system poses the questions which are required to accomplish the task. To handle this complexity, a huge technical effort is necessary, concerning the dialogue manager as well as pre-processing modules.

The dialogue management has to *verify* the recognition engine's hypotheses about the user's utterances. The most primitive way is to explicitly ask for an acknowledgment after each input. A much more natural dialogue flow can be achieved by one single acknowledgment of all data in the end of the dialogue. Of course then, the user has to specify in an intermediate step which piece of information has been recognized wrong. The more natural input the dialogue system allows, the more flexible the *verification* strategy can be handled.
To enable a successful and natural dialogue, the dialogue manager requires *knowledge sources*. Referring to [5], these sources are called dialogue model. The model might consist of different types of knowledge sources: a dialogue history, a task record, a world knowledge model, a domain model, a generic model of conversational competence and a user model.

3. DESIGN AND INTEGRATION

The aim of the design of the developed dialogue manager is to extend the framework introduced in 3.3 by providing natural spoken dialogues and a new approach to driver information, the so called information assistants.

3.1 INTELLIGENT ASSISTANTS

One main topic is the situation- and user-aware presentation of information, the multimodal in- and output, as well as the cross-linking of functions. One approach to these issues is the idea of so called information assistants. They are introduced to offer an optimal situation-dependant support to the driver. She or he should efficiently be led through dialogues matching her or his current needs.
Situation-awareness refers to the consideration of external influences. The dialogue between user and system has to be context-dependent (see chapter 1.2). The assistants can be initiated by the user in case he has any needs, as well as by the system, for example, if the car is running out of gas. Other examples for this idea of assistants are a restaurant guide automatically considering the drivers preferences and an end-of-journey-assistant which looks for a parking lot and transmits a map of the surrounding area to the PDA when the driver arrives at the desired destination.

3.2 IN- AND OUTPUT

The dialogue manager is designed to process *input* in terms of intentions. This means, the output string of the recognizer is preprocessed in terms of semantic interpretation. Accordingly, tactile input as well has to be preprocessed.
This generic intention based input processing enables to plug arbitrary input modalities and their preprocessing module onto the framework. Multimodal interaction is assured. While running a dialogue, the input modality can freely be changed.
A well-designed spoken dialogue per se provides assistance to the driver because there is no need to turn his visual focus from the street to a display.
The frame-based approach for dialogue management is implemented. It provides adequate flexibility while keeping technical requirements manageable. In this case, flexibility means user's flexibility to freely decide how much information to submit in one step. By this, an expert user can reduce the dialogue run, whereas a novice user still can be led through the dialogue step by step. This flexibility as well reduces handling errors because there is no preassigned sequence in which the information slots have to be filled.
When we speak of natural language in this context, we talk about a spoken input where words without semantic relevance may appear. The system is able to process inputs

like for example “Well... Please take me to Arcis Street 16 in Munich and take the fastest way.” The semantic interpretation is done by a module called EASY discussed in chapter 3.3.
As already mentioned, the dialogue shouldn’t be controlled only by spoken input, but by other modalities as well. To allow for tactile interaction, a module had to be developed to determine the user’s intention by relating tactile input with the current state of the displays.

The dialogue manager’s *output* is freely configurable with respect to the message pattern used in our framework (see 3.3). There are three points in the dialogue flow where an output can be defined. First, when successfully finishing a dialogue, messages including user’s input information can be sent, for example, to a service, like a navigation system. Further more, it is possible to put out messages after having processed a user input. This can for example be applied to give feedback about the dialogue progress. The third point is the system requesting an input if necessary.
Spoken feedback is synthetically generated in our setup. The text to be spoken is sent to a TTS-server via a HTTP-request (see 3.3).
One basic principle of usability design is that in any situation, the user has to know the state the system is in. To ensure this, a color-coded speechy-symbol is shown in the head-up display. When finished, the recognized intentions are displayed for three seconds. In case of an error, the user can understand that the malfunction of the system is due to a recognition error and can correct this. In case the confidences of the speech recognizer stay below a customizable threshold, a red speechy-symbol is shown to signalize an error. A prompt will then be repeated more verbosely.

3.3 FRAMEWORK

The framework is organized in form of a client-server structure, consisting of an input, a fusion, and an output layer. The input layer consists of all kinds of input devices (like a speech recognizer or a button array). In the output layer, there are application modules, like a navigation system or an MP3-player. The core unit is the multimodal integrator. All modules of the input and the output layer must register as clients at the database of the integrator. The communication between input, fusion, and output layer is realized via a bidirectional exchange of string messages streamed over TCP/IP-connections (socket backports).
Using a look-up table, a meta-device (the so-called command mapper) converts all messages of the single recognizers. The mapping process is based on the formalism of a context-free grammar. Thus, the proprietary message output strings of the individual recognizers are formatted into a standardized device-independent structure. The strong modularization allows for a fast and straightforward replacement and an integration of additional modules. Consisting of several networked components, the integrator interprets the multimodal message stream that is continuously arriving from the individual recognizers.
The input of the recognizers is combined via Late Semantic Fusion (LSF). A string parser checks the messages for syntactical correctness and for integrity. A finite state machine provides and manages the database for the multimodal integration process. In this process, secondary knowledge is included from the application module and the integration status. The intention decoder forms the central component within the multimodal integrator. Considering additional context information (see 1.3) the messages are evaluated via a semantic unification process. The result is checked by a set of additional components (e.g. an error manager). Finally, the integration unit generates a device-independent command which is, analog the lines of above, transformed in a proprietary format of the application modules. If, for any reason, the resulting command can not be applied in the current system context, or is incorrect, the dialogue manager generates a dedicated error dialogue.
The following modules were used to integrate the dialogue manager into the existing framework.

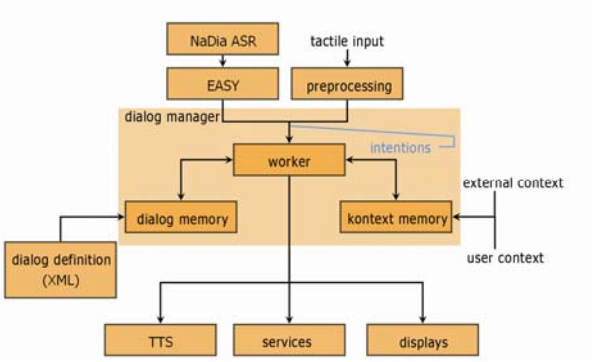

Figure 1: Dialogue Manager’s Architectural Overview

Our experimental setup uses an *ASR* called ODINS developed by our institute [6]. This speaker-independent recognizer is based on intraword triphone HMMs. The phoneme models were trained with a corpus containing spontaneous speech utterances from the Verbmobil-project [7].
As input, one has to provide a speech model, a grammar which contains all possible permutations of words as a lattice-network, ideally with probabilities for each word transition. Second, it requires a thesaurus, a collection of all possible words, and one or more phoneme representations of it. ODINS uses the Sampa phonem list.
The recognizer has to be activated manually, for example, by pressing a push-to-talk-button. The recognizer output is an n-best-list containing five sentences with the highest sentence confidence, together with single word confi-

dences, which are the logarithmized probabilities for each word.
In order to be used by the dialogue manager, the results of the ASR have to be *interpreted semantically*. This is done by another institute‘s development called EASY [8].
It basically tries to determine user’s intentions out of the utterance detected by the recognizer. For example, a spoken input like “I’d like to drive to Munich, to Arcis street, please.” would lead to two intentions “destination_city: munich” and “destination_street: arcis street”.
EASY uses Bayesian Belief Networks to match the recognized words to one or more predefined user-intentions. To perform this, it requires of course a source of knowledge which defines the composition of the Bayesian Network.
Its output, the intentions, are rated by confidences. These are used by the dialogue manager to weigh the input intentions.
The *speech output* in our setup is synthesized by a server running AT&T Natural Voices. The output-string to be synthesized is sent via a HTTP-request to the server which returns a WAV-file.
To be able to use tactile input devices, an additional module has to be designed which pre-processes the input commands. Furthermore, this module is responsible for controlling the displays. The dialogue manager and the dialogue definition should be kept free of display-specific commands. The desired command flow is a meta-command like “show restaurant search“ by the dialogue manager which is received and interpreted by the display control. It accordingly sends out the display specific commands. This allows different behaviors on different displays. A central information display realized as touch screen, for example, could show a map with restaurants nearby while in the HUD, these restaurants can be shown as a vertical list.

We use *XML* to specify the dialogue. Each dialog has an arbitrary number of subdialogues; each of them has a context condition assigned. The subdialogue-node has two kinds of children: A send-element in which socket messages can be defined and the current information frames.
They are associated with intentions coming from the preprocessing input modules. This association is one basic part of the dialogue definition. By this, the appropriate dialogue is figured out and the user’s information is processed. A “required”-attribute differs mandatory and supplementary information frames. Only mandatory frames are enquired, optional ones have to have a default value assigned.
Each frame may have different “inquiry”-children which contain information about how to prompt for information to fill this frame. Every inquiry has a “verbose”-value assigned to. Thus, differently detailed prompts can be defined and used by the dialogue manager to enable context adaptation and error management.

3.4 CONTEXT PROCESSING

One main focus in the development of the dialog manager was the ability to make context-dependent decisions. As stated in paragraph 1.2, different context parameters have an impact on the driver.
Our dialogue manager provides three possibilities to affect a dialogue. It is capable of varying the dialogue itself, as well as initiating a dialogue as a response to a certain situation. Varying the dialogue means, for example, changing the verbosity of the dialogue output with respect to the user’s knowledge about the system. The initialization of a dialogue would be reasonable if, for example, the car is running out of gas. The third way of using context information is to retrieve dialogue input out of it. An example would be the actual city in which the car is placed. This could be used as information for a navigation dialogue which the user doesn’t need to provide.
The current version of the dialogue manager uses a rule-based approach for context processing. Context variation is done by defining an arbitrary number of subdialogues where each subdialogue is valid for a dedicated context condition.
Dialogue initialization works quite similar. For each dialogue an initialization condition can be set. If no other dialogue is currently active and one of these conditions becomes true, its specific dialogue will be started.
Dialogue information retrieval is possible because each incoming context value is stored. Out of this memory, desired values can be fetched.

4. EVALUATION

The designed and implemented dialogue manager has been evaluated in a usability experiment in our institute’s driving simulator. The goal was to analyze the usability of the system for first-contact users as well as the choice of input modality. The test persons were asked to handle two of the assistants mentioned in chapter 3.1, but fo focus mainly on their driving performance. The assistants could be utilized by natural speech and via a controller.
The evaluation was divided into two parts. At the beginning, the task was to enter a navigation address by speech. In the first step, we let the user decide which input strategy to embark to find out the user’s intuitive way to deal with this speech interface. The second experimental part introduced the assistants. While driving, the fuel assistant was started and informed the driver about the empty tank. The user was expected to successfully run through the assistant, and to choose one of the suggested gas stations.
Afterwards, the test person should notify the system about being hungry using his own words. This fuzzy input started another assistant, a restaurant search. The test per-

son could handle these two assistants by speech as well as by a keypad and a touch screen.
After these tasks were accomplished, the user's subjective opinion has been acquired by a questionnaire.

4.1 RESULTS

We acquired 22 subjects, 15 male and 17 female. The mean age was 37.7 years. Half of them stated they have already had experience with speech recognition systems, mainly in the telephony context.
The first task of our experiment indicates a tendency towards a complete input of the navigation destination. 14 of the 22 subjects intuitively used this strategy, eight chose the iterative menu-based way.
After having experienced the two input possibilities, 15 persons now have used the one-step strategy. At a closer look, it emerged that out of the seven people at first using the iterative speech menu, none had retained this strategy after getting to know the possibility of a complete input in one step.
In the interview afterwards, the participants rated the possibility of entering more than one piece of information at once as very important and comfortable. The analysis of the time and steps the subjects needed to accomplish the tasks revealed an evident result. The possibility to enter all needed pieces of information at once required only 55% of the time using a speech menu for the same task, and thus, it was much more efficient.
The utterances while using the assistants in the second part of the evaluation have been from 100% (yes-or-no question) to 50% (free expression of hunger) command-based.
While entering a navigation destination in the iterative way, 91% of the spoken input was command-based. The complete input at once had a ratio of 77%. The possibility to use natural language has not been used as much as expected regarding to the online survey results (see 1.4).
While using the assistants, spoken and tactile input were equally used. Interestingly, no significant distinction between different age groups could be determined.
The assistants were widely accepted. On a scale from 1 (which was the best) and 6 (as the worst grade), the fuel assistant was rated 1.62 and the restaurant search 1.95. A system initiation was only desired to evade distress, for example, only at very little gas left.
The largest problem for the test subjects was the intuitive operation of the PTT-key. The system expected the user to press the button once and shortly before every utterance. Only three of the 22 subjects did this correctly. A much larger part, six of them, kept the button pressed during the whole input like using a walkie-talkie. The most frequent way to use the button was to press it once to start a dialogue but not to press it if the system poses a question.

5. CONCLUSIONS

In this article, we gave an introduction to the automotive domain in terms of usability and presented a possible classification for spoken dialogue systems.
We discussed the design of our context-aware dialogue manager and its integration into the existing framework.
A usability evaluation revealed an increase in efficiency and joy of use, enabled by the frame-based approach.
In ongoing and future work, the integration of several multimodal combinations of input and output devices with the dialogue manger is going to be implemented and tested.
As well, one might evaluate a statistical approach to adapt dialogue parameters in reaction to context influences. Future research might as well deal with an agent-based dialogue manage-ment strategy.

5. REFERENCES

[1] Rasmussen, Skills, Rules and Knowledge.
In: IEEE Transactions, SMC-13 (1983), S. 257 – 266

[2] Donges, E., "Das Prinzip Vorhersehbarkeit als Auslegungskonzept für Maßnahmen zur Aktiven Sicherheit Maßnahmen zur Aktiven Sicherheit." In: Das Mensch-Maschine System im Verkehr, VDI-Berichte (1992), Nr. 948

[3] Oviatt, S. et.al. "Error Resolution during Multimodal Human-Computer Interaction", ICSLP 1996, Philadelphia, in: Proc. Vol.I

[4] McGlaun, G. et.al., "Kontextsensitives Fehlermanagement bei multimodaler Interaktion mit Infotainment- und Kommunikationseinrichtungen im Fahrzeug." Tagungsband VDI-Fachtagung USEWARE 2004, 22.-23.06.2004, Darmstadt. VDI-Bericht 1837 "Nutzergerechte Gestaltung technischer Systeme"

[5] McTear, M., "Spoken Dialog Technology: Enabling the Conversational User Interface". In: ACM Computing Surveys 34 (2002), Nr. 1, S. 1–80

[6] Thomae, M. et.al., "A One-Stage Decoder for Interpretation of Natural Speech. Institute for Human-Machine Communication", Technische Universität München. 2003

[7] Wahlster, W. Verbmobil: "Foundations of Speech-to-Speech Translation", Springer, Berlin, 2000

[8] Schuller, B. et.al., "Multimodal Music Retrieval for Large Databases", ICME 2004, IEEE Int. Conference on Multi-media and Expo, Taipei, Taiwan

[9] Althoff, F. et.al., "Towards a New Approach for Integrating Mul-timodal User Input Based on Evolutionary Computation." Proceedings ICASSP 2002, Orlando, USA, 13.-17.05.2002. IEEE Signal Processing Society. CD-ROM. (PDF)