Task Hierarchies Representing Sub-Diags
in Speech Dialog Systems

Ute Ehrlich, DaimlerChrysler, Research and Technology
ute.ehrlich@daimlerchrysler.com

ABSTRACT
The application of a speech dialog system is modeled as a hierarchy of sub-tasks in order to
• have the possibility of structuring complex dialogs into sub-diags and thus restricting the lexicon to be recognized at each dialog state,
• to model possible sequences of sub-diags,
• to enable several instances of the same sub-tasks,
• to make references and/or switches to prior discussed sub-tasks possible.
Thus, also more complex applications can be modeled and handled without losing speech recognition accuracy or getting lost in different dialog themes. As these restrictions are quite natural users will not perceive any limitation.

Keywords: speech dialog, dialog control, task hierarchy

1. INTRODUCTION
The acceptance of currently available dialog systems realized with IVR (Interactive Voice Response) technology is not very high if long question-answer cascades are needed to get the required information. A far greater flexibility of the dialog has to be ensured. The dialog control of the DaimlerChrysler Speech Dialog System offers such a flexibility determining system reactions by selecting and achieving dialog goals using a user adaptive dialog strategy (see e.g. [1], or systems with alternative approaches, e.g. [2], [3]). If dialogs become more complex handling also different themes within one dialog, the flexibility has to be restricted. This can be done adding a task based control to the purely dialog goal driven control (see e.g. [4]). The approach described in [5] relies on the modeling of sub-diags. In this system possible dialog sequences are rather fixed. Therefore we choose another approach modeling the dialog application as a task hierarchy where the required flexibility within and also between selected tasks is still provided. So, we present an approach for the dialog control which restricts the dialog flow in a natural way, offering also the possibility of restricting the recognition lexicon at each dialog state. But it also keeps the dialog flexibility as far as necessary – and possible.

2. DIALOG CONTROL IN THE DAIMLER-CHRYSLER SPEECH DIALOG SYSTEM
The DIALOG COMPONENT is the main component of the speech dialog system (see e.g. [6]). It controls the whole dialog, i.e. initiates analysis and generation processes of the other modules (SPEECH RECOGNIZER, LINGUISTIC INTERPRETATION and GENERATION/synthesis) based on the input utterance and the current dialog context. It also decides which information, questions, or explanations are given at what time to the calling persons.
The dialog component comprises three parts:
• the CONTEXTUAL INTERPRETATION relating the utterance to the dialog context,
• the proper DIALOG CONTROL determining the dialog continuation based on active dialog goals and a flexible dialog strategy,
• the DIALOG CONTENTS interpreting and predicting the contents of utterances in the application context.
In the following we concentrate on the DIALOG CONTROL and the DIALOG CONTENTS.
The DIALOG CONTROL is responsible for the overall control of the dialog, i.e. it determines the next system initiative or reaction. This determination is based on the selection of one or more active goals to be achieved. Dialog goals are generated based on the utterance interpretations and the current dialog strategy. Goals are added, their state is changed, or they succeed as a result of the interpretation of the user utterance. So the CONTEXTUAL INTERPRETATION causes confirm goals (for confirming the understood information), whereas the DIALOG CONTENTS causes goals for requesting a missing parameter value, for informing about a found result or assumed defaults, and for explaining problems or inconsistencies.
Which goals are chosen for achievement is dependent on the current dialog strategy. The dialog strategy allows a combination of the confirmation of one or several parameters with or without a system initiative in one system turn or confirming yes/no or the spelling mode if something goes wrong. The strategy is dynamic in that the system adjusts its setting according to whether it encounters contradiction from the caller or not.
The DIALOG CONTENTS has the control about the contents of the dialog: it determines the sequence of the parameters needed for executing the caller’s task, determines default values for parameters dependent on
As we have shown above, the dialog control decides how the system reacts to a user utterance based on current dialog goals and a local strategy. So far the dialog contents is not taken into account. This, of course, is not applicable for more complex applications or dialogs handling different tasks. If for example, a user wants to go from one place to another using or the train and buses – here he needs the time tables, maybe a ticket, and a reservation – or he prefers going by car asking the system for the best route and maybe the reservation of a parking lot. In this example it is not possible to let the user speak always about everything, i.e. jumping from one theme to another. We have to structure the dialog into possible sub-tasks (e.g. time table information about trains, about buses, train tickets, train reservation, route planning for cars, reservation of a parking lot). This is necessary for different reasons:

- Also for humans it would become quite complicated to follow such jumps. Misunderstandings are inevitable.
- The user of a speech dialog system should have the impression that the system has a clear concept of the dialog. He would not understand if the system would jump between themes, especially if the jumps are forced by recognition errors.
- Due to word recognition limitations – or due to the fact that the word recognition becomes more reliable if the size of the recognition lexicon can be reduced, it is important to define smaller lexical parts for each dialog sub-task. Of course, this prevents the system from understanding everything at every time. Therefore the modeling of the tasks and their respective lexical parts must be done carefully in order to ensure natural switches between themes.

This structuring of dialogs into sub-dialogs does not mean that the dialog now follows a strict dialog model (e.g. given by a flow chart). Within the currently activated theme or themes the dialog control determines the dialog continuation based on dialog goals and strategy as described above.

Additionally, with the concept of several sub-tasks modeling dialog themes it is also possible to talk about different instantiations of the same theme (e.g. several train connections or different routes to one goal) and to take up again prior discussed themes using a kind of meta-language, e.g. ‘*Let’s come back to the first route.*’ So, we model the application as a hierarchy of sub-tasks in order

- to have the possibility of structuring complex dialogs into sub-dialogs,
- to model possible sequences of sub-dialogs,
- to enable several instances of the same sub-task (e.g. ‘*Is there another train connection*’ ...),
- to make references and/or switches to prior discussed sub-tasks possible.

![Fig. 1: Hierarchy of Task Types](image-url)
4. STRUCTURING A DIALOG INTO A SUB-DIALOG HIERARCHY

Each application is modeled as a task type hierarchy (see e.g. Fig. 1 for the above described example application), where each sub-task has a set of task parameters modeling the information relevant for the respective task (or theme). Each task type with child task types has a special CHOICE parameter, which enables the switch between the different child task types (e.g. Fig. 3, User 5: ‘... How about using the car?’ ⇒ task type car). The tasks ‘inherit’ the task parameters of their parents. E.g. in Fig. 1 the parameters source and goal are relevant not only for the task type travel, but also for train (connection) or seat Reservation. The task type hierarchy determines the possible sequences of tasks, i.e. sub-dialogs discussing a specific theme. The dialog always starts with a task of the root task type (here identification). All parent tasks of the current task are always activated, e.g. if the current task is of type travel, also task parameters of all parent tasks (task types train and identification) are available. If the task parameters of a task should not be changed when they are confirmed once (e.g. the parameters name and password should be given only once), the task type gets
the feature \textit{CLOSED} meaning that after the confirmation of all relevant task parameters the task (here \textit{identification}) is closed. Then the parameter values are available e.g. for database requests, but it is no longer possible to talk about them. After a new user utterance the \textit{ADMISSIBLE} task types for the next current task are determined in the following way (see example in fig. 2 and fig. 3):

1. The current task remains current (e.g. \textit{train/1}).
2. The task type of the next current task is a child or (great)grandchild task type (e.g. \textit{public\_transport}). The new current task can be a new task instance, or the re-opening of a completed one using a meta expression (e.g. User 7: ‘... I \textit{will take the train ...’}).
3. If the current task is complete, i.e. all obligatory task parameters are given and confirmed, it will be closed. The parent task becomes the new current task. If it is closed, the only parameter to talk about is the \textit{CHOICE} parameter. Also here it is possible to jump to a grandparent, closing all the tasks on the way up.

This means that the direction in the hierarchy must be preserved. It is not possible going first up and then down or vice versa within one step (e.g. from \textit{train/1} to \textit{car/1}).

Which task will become current depends on the task parameters detected in the user utterance. The task parameters are attached to task types. Thus, the occurrence of task parameters determine a set of possible task types. Out of these the one is selected:

1. If the task types are not lying on one path in the task type hierarchy the common parent with the greatest distance to the root is the next current task. The parameter values which cannot be attached to a task are stored and evaluated later.
2. A new task with an admissible type and the greatest distance to the root becomes the new current task, if the task type of the current task lies on the path from there to the root.
3. The current task remains current if the task type with the greatest distance to the root lies on the path from the task type of the current task to the root.

The admissible task types restrict also the possible task parameters specifying the expected sub-language of the next user utterance. Thus, it is also possible

1. to restrict the system’s recognition lexicon,
2. to restrict the context for the CONTEXTUAL INTERPRETATION.

The above described selection of admissible task types can be influenced by a global strategy:

- Rigid strategy: only one task can be active at a time (e.g. the \textit{identification} of a person must be finished before any other information is accepted). This strategy is recommended for increasing the reliability of the system, especially if the word recognition is a problem due to e.g. a bad telephone line quality or large recognition lexica.
- The above described open strategy.

The described algorithm for processing the task hierarchies is application independent, so that for each new application - or a different system behavior - only the task hierarchy itself must be modeled, which is a fairly simple job.

5. CONCLUSION

The dialog control in speech dialog systems usually is modeled as a kind of a flow chart determining the possible follow ups at each dialog state. The DaimlerChrysler Speech Dialog System dialog control selects system reactions by achieving goals following a user adaptive dialog strategy locally without having a globally fixed ‘plan’ of the dialog structure. But this means that nearly every user utterance of the application context can be interpreted at every time. With dialogs becoming more complex it is no longer possible to allow always the whole application context. This would enable puzzling jumps between dialog themes, which is not desirable. And the lexical size of such a complex application would become too large to ensure the recognition reliability.

We showed a method of structuring the dialog into different sub-dialogs representing different dialog themes. This adds a kind of a task dependent dialog initiative to the local dialog control, reducing the dialog complexity at every dialog state, but guaranteeing also the flexibility of the dialog according to human expectations.

The method is fully implemented in Prolog (with the exception of re-opening tasks by referring to them using meta expressions) and was tested successfully with dialog applications similar to the described example.

6. REFERENCES