BRIDGING THE GAP BETWEEN MIXED-INITIATIVE DIALOGS AND REUSABLE SUB-DIALOGS

Susanne Kronenberg, Peter Regel-Brietzman

Speech Understanding Systems
DaimlerChrysler AG

ABSTRACT

For easing the developing process for dialog systems it is desired that reusable dialog components provide pre-packaged functionality 'out-of-the-box' that enables developers to quickly build applications by providing standard default settings and behavior [6]. Additionally, human-computer interaction should become more human-like in that mixed-initiative dialogs are supported. Mixed-initiative interaction requires the system to react to user initiated application specific commands whereby reusable dialog components have to be application independent to be used in different settings. This article presents a dialog mechanism, so called *meta-dialog*, which is responsible for the control flow between reusable sub-dialogs and mixed-initiative dialogs.

1. INTRODUCTION

Recently, the use of speech based dialog systems has became more and more widespread in most application areas, e.g. call-center applications. By now for each specific application, e.g. cinema preview or online banking, different dialog-systems are being developed. However, the fundamental dialog strategies are often very similar and only the application specific parts of the systems are different. In order to ease the developing process, dialog systems should be implemented by reusing similar parts of the system for different applications and only accommodating those parts of the system, which are application specific. This can be realized based on reusable sub-dialogs. These reusable sub-dialogs are components which are required not to change significantly between application or vendor. Therefore, for these components, both component-specific configurable parameters and the return semantics have to be specified which in addition have to be application independent.

On top of that, it is desirable for human-computer interaction to become more human-like. Many researchers have noted that the absence of mixed-initiative gives rise to two problems with expert systems: They do not allow users to participate in the reasoning process, or to ask those questions they want to have answered [1], [4]. Accordingly, mixed-initiative dialogs have to be integrated to achieve advanced human-computer interaction. In mixed-initiative dialogs the user is free to determine the direction of the dialog flow at each step. Moreover, in cases, in which the user takes the initiative of the dialog the predefined dialog flow is changed by the user because only in those cases the user's initiative is necessary. These changes can be assumed to be application specific in order to establish human-computer cooperation in such a way that the system is able to understand the user's utterances[1].

[1]Otherwise, these user initiated utterances will be

This implies that the dialog flow of reusable sub-dialogs can also be changed by application specific commands. But this fact contradicts the requirement as stated above, namely that reusable sub-dialogs are not application specific.

This article presents a dialog mechanism, so called *meta-dialog*, which will fill the gap between the requirements of reusable sub-dialogs and mixed-initiative dialogs. Therefore, the meta-dialog of an application changes application independent components into application specific ones.

2. META-DIALOGS

2.1. Mixed-Initiative Dialogs versus Reusable Sub-Dialogs

The aim for designing dialog systems by using reusable sub-dialogs is that the main part of the dialog will be built based on these components and only the application specific parts have to be handcrafted. The application specific part of the dialog consists of the dialog flow, i.e. the order, in which the components will be processed, the application specific parameters, and the application specific vocabulary. This implies that the input of a reusable sub-dialog will be application specific and the result of a reusable sub-dialog has to be interpreted according to the application specific requirements to maintain the predefined dialog flow.

Moreover, reusable sub-dialogs have to be able to understand application specific commands. For example, a browsable selection list allows the user to hear items in a list in sequence and navigate through this list. Any item may be selected at any time by saying the appropriate grammar entry. Consequently, specific commands are used for selecting a list entry, which corresponds to the different domains. In an online banking application a money transaction can be selected by, e.g. ‘transferring,’ whereas in a news reader application it is more likely that a topic will be selected by ‘reading.’ Accordingly, the application dialogs have to transfer a reusable sub-dialog, which is application independent into an application specific dialog, which shows the reactions, which are required in this special setting.

In [7] it is shown that control shifts do not always correspond to task boundaries. For example, in a browsable selection list the user can use application specific commands different to the ones for selecting a list item or for navigating through a list. Mostly, this control shift will lead to a topic shift, i.e. the user initiates a sub-dialog different to that of a browsable selection list. Therefore, by using such commands the user takes control over the dialog flow and will force the system to change the dialog flow, which in turn can imply that the special sub-dialog will be terminated.

But in [8] it is shown that changes of control can occur without a topic shift. For guaranteeing that not every control shift leads to a termination of a sub-dialog it is necessary that application specific commands have to be known to the reusable sub-dialogs and the dialog flow has to be changed, if necessary in accordance to these commands.

As reusable dialog components are required to be application independent these changes in the dialog flow cannot be handled by these components themselves. These changes will be determined by the overall dialog control, so called *meta-dialog*. The purpose of the meta-dialog belonging to an application dialog is twofold:

1. First, the meta dialog has access to the lexicon, which contains the application vocabulary. If the application task has an hierarchical structure, the lexicon has to reflect this structure. For instance, in a news reader application a news item can be selected by the user by naming first the news category, i.e. ‘politics,‘ ‘sports,‘ etc. and then selecting a special topic like ’tennis.’ Furthermore, the user can directly ask for tennis news. Accordingly, the lexicon has to be structured in such a way that ’tennis’ is subsumed by ’sports,’ which means that

treated by the error handling component of the system.

the system can directly access this topic by knowing that 'tennis' is a special topic of the 'sports' category. Otherwise, not every application specific command has to be known by each reusable sub-dialog. A confirmation dialog should not have access to every lexicon entry in order to avoid overgeneralizations, by which also misrecognized utterances will always lead to a topic shift. Therefore, the meta-dialog guarantees that every sub-dialog has access only to that part of the lexicon, which is necessary for supporting mixed-initiative.

2. Second, the meta-dialog controls the dialog flow. Every application dialog has a predefined dialog flow which is followed in case no initiative is taken by the user and then control over the dialog is left to the system. If the result of a reusable dialog indicates that the control is taken by the user the meta-dialog will change the dialog flow according to the user's initiative.

2.2. The Processing Strategy

The presented processing strategy for meta-dialogs is based on two theories of initiative strategies. The adaptive model as introduced in [2] merges discourse structure and accounts of initiative in that initiative is held by the discourse segment initiator. Each discourse segment has a purpose and the purpose of each segment contributes to the purpose of its parents. The hierarchy of segment purposes makes up the intentional structure. Intentional structure is the key to understanding what the discourse is about and explains its coherency. In [5] each discourse event is explained as either: (i) starting a new segment whose purpose contributes to the current purpose, (ii) continuing the current segment by contributing to the current purpose, or (iii) completing the current purpose. In this strategy, called *Collagen*, each participant maintains a mental model, called discourse state, of the status of the collaborative tasks and the conversation about them. A discourse state consists of a stack of goals, called the *focus stack*, and a plan tree for each goal on the stack. The top goal on the focus stack is the 'current purpose' of the discourse.

The underlying concepts of both theories are used to model mixed-initiativess in combination with reusable sub-dialogs. Participants in a collaboration derive benefit by polling their talents and resources to achieve common goals. In a task oriented dialog the common goal is to complete a special task. For achieving this goal several task parametes must be set. Accordingly, a successful dialog will be one, in which all task parameters are set. Furthermore, by considering the reusable sub-dialogs as discourse segments it is required for initiatives to be able to change during processing of a discourse segment different to the theoretical frameworks introduced above.

As mentioned in (2.1) the application dialog is augmented with a predefined dialog flow, i.e. a predefined sequence, in which the task parameters will be processed. This sequence of task parameters can be considered as the focus stack. In case the predefined dialog flow is followed the focus stack will be sequentially processed and the top of the focus stack contains the currently considered task parameter. In case initiative is taken by the user and the dialog flow is changed, the task parameter corresponding to the user command is considered as the currently considered one, i.e. the current top of the stack. Every task parameter is associated to a discourse segment, i.e. the part of the application dialog, in which this parameter will be processed. For reaching the entire dialog goal each parameter has to be set. Thereby, each discourse segment contributes its purpose, i.e. the corresponding task parameter, to the purpose of its parents, i.e. the discourse corresponding to the already processed discourse parameters. If the predefined dialog flow is followed, this contribution will lead to a completion of the the current dialog goal. In case the predefined dialog flow is not followed, i.e. control is taken by the user, this contribution can either complete or correct the current dialog goal. Completion of the dialog goal is given if the predefined dialog flow is changed but a task parameter is processed which has not been considered before. Correction of the dialog is given, if the predefined dialog flow is changed but a task parameter is processed which has already been considered. Accordingly, as the meta-dialog is responsible for keeping track of the dialog flow the meta-dialog has to determine which kind of contribution to the current dialog is given by a new incoming task parameter.

This decision process is based on two discourse relations, called *complete* and *correct* (cf. [3]), which enable the system to process both kinds of contributions. In case a task parameter is considered for the first time the complete relation will add the information given by this parameter to the discourse purpose. The dialog will be continued by the next task parameter – the new top of the focus stack – which has not yet been considered. If a task parameter has already been considered and is visited for the second time the correct relation will update this task parameter. The dialog will be continued by the next task parameter on the focus stack which has not yet been considered. Based on both discourse relations each task parameter is processed, whereby both cases, i.e. initiative taken by the user or the system, are covered by this discourse framework.

Accordingly, the presented initiative strategy is based on tracking discourse structure. The focus stack reflects the predefined dialog flow whereby focus is given to the top element of the stack, i.e. in cases, in which control is left to system. In cases where initiative is taken by the user, the new top of the focus stack corresponds to the currently considered task parameter.

3. CONCLUSION

This article presents a dialog framework, called meta-dialog, which integrates the requirements of reusable sub-dialogs and mixed-initiative dialogs into one processing strategy. This processing strategy is based on a focus stack and on tracking the discourse structure. Whereby the focus stack reflects the dialog flow. Two discourse relations are responsible for maintaining the discourse structure, which reflects the control flow between user and system in order to achieve the final dialog goal.

4. REFERENCES

[1] David M. Frohlich and Paul Luff. Conversational resources for situated action. In *Proc. Annual Meeting of the Computer Human Interaction of the ACM*, 1989.

[2] P. Heeman and S. Strayer. Adaptive modeling of dialogue initiative. In *Proceedings of the NAACL Workshop on Adaption in Dialogue Systems*, pages 79–80, Pittsburgh, USA, June 2001.

[3] Susanne Kronenberg. *Cooperation In Human Computer Communication.* PhD thesis, Bielefeld University, 2001.

[4] Martha Pollak, Julia Hirschberg, and Bonnie Webber. User participation in the reasoning process of expert systems. *AAAI82*, 1982.

[5] Charles Rich, Candace L. Sidner, and Neal Lesh. Collagen: Applying collaborative discourse theory to human-computer interaction. *AI Magazine, Special Issue on Intelligent User Interface*, 2001.

[6] W3C. *Reusable Dialog Requirements for Voice Markup Language*, April 2000. http://www.w3.org/TR/reusable-dialog-reqs.

[7] Marilyn Walker and Steve Whittaker. Mixed-initiative in dialogue: An investigation into discourse segmentation. In *Proceedings of the 28th Annual Meeting of the Association of Computational Linguistics*, pages 70–78, 1990.

[8] Steve Whittaker and Phil Stenton. Cues and control in expert client dialogues. In *Proceedings of the 26th Annual Meeting of the Association of Computational Linguistics*, pages 123–130, 1988.