1 Introduction

As information services and domains grow more complex the complexity of dialogue systems increases. They tend to need more and more domain knowledge and the domain reasoning mechanisms also have to become more sophisticated. Utilising domain knowledge reasoning is in many cases necessary for a dialogue system to interpret and respond to a request in an intelligent manner, especially as requests can be vague and sometimes ambiguous. This involves not only requests for information from application specific knowledge sources, but also requests related to the properties and structures of the application and requests that are outside the scope of the application. Thus, dialogue systems must be able to access, gather and integrate knowledge from various domain knowledge sources and application systems in order to determine the precise meaning of a request and produce an appropriate response. However, although the dialogue system gather information from various sources we assume that the tasks are well-defined and that the users have articulated information needs that they can express in specific terms.

2 Dialogue and domain knowledge management

Dialogue systems often have a modular architecture with processing modules for interpretation, dialogue management, background system access, and generation. The processing modules utilise a number of knowledge sources, such as, grammar, lexicon, dialogue model, domain model, and task model. In this paper focus is on dialogue management and domain knowledge management, which includes background system access (for more details see Flycht-Eriksson and Jönsson (2000)).

The role of the Dialogue Manager differs slightly between different dialogue system architectures, but it's primary responsibility is to control the flow of the dialogue by deciding how the system should respond to a user utterance. This is done by inspecting and contextually specifying the information structure produced by an interpretation module. If some information is missing or a request is ambiguous, clarification questions are specified by the Dialogue Manager and posed to the user. Should a request be fully specified and unambiguous the background system can be accessed and an answer be produced. As a basis for these tasks the Dialogue Manager can utilise a dialogue model, a task model, and a dialogue history.

The Dialogue model holds a generic description of how the dialogue is to be constructed, i.e. to decide what action to take in a certain situation. It is used to control the interaction, which involves determining: 1) what the system should do next (and what module is responsible for carrying out the task) and 2) deciding what communicative action is appropriate at a given dialogue state.

The System Task model represents how the system's tasks are performed. However, the terms task and task model can refer to very different phenomena. It is important to make a clear distinction between the system's task(s) and the user's task(s). A user task is non-linguistic and takes place in the real world. Models of such tasks involve the user's goals and how they can be achieved. Models of system tasks describe how the system's communicative and other tasks, e.g., database access, are carried out.

A typical example of the difference between the two types of task models can be found in a timetable system where the user states that (s)he needs to be at the train station to catch a certain train and requests information on buses going there. The information that the user is going to the train station is user task model information, indicating that buses arriving after the departure time of the train are not relevant. The system task model on the other hand models the information required for complex requests, such as date and departure place in a time-table system. It is used by the Di-
dialogue Manager when collecting user information in order to perform a background system access.

The Dialogue history records the focus of attention and contains information about objects, properties, and relations as well as other dialogue information such as speech act information and system task information.

To develop a Dialogue Manager that easily can be customized to new domains and in which different dialogue strategies can be explored, the Dialogue Manager should only be concerned with phenomena related to the dialogue with the user. It should not be involved in the process of accessing the background system or performing domain reasoning. These tasks should instead be carried out by a separate module, a Domain Knowledge Manager.

The Domain Knowledge Manager is responsible for retrieving and coordinating knowledge from the different domain knowledge sources and application systems that constitutes the background system. The Dialogue Manager can deliver a request to the Domain Knowledge Manager and in return expects an answer retrieved from the background system. If a request is under-specified or contains inconsistencies from the Domain Knowledge Manager's point of view, a specification of what clarifying information is needed will instead be returned to the Dialogue Manager.

Although fully specified, requests can contain vague or ambiguous information or even some errors that can not be detected and handled without extensive domain knowledge. This type of domain knowledge is stored in the domain knowledge sources. They contain knowledge of the world that is talked about and can vary much in form and content. Information from a domain knowledge source is primarily used to find the relevant items and relations that are discussed, to supply default values, etc. The knowledge represented in a domain knowledge source is often coupled to the application system, e.g. a database system. In such cases it is often used to map information from a Dialogue Manager to concepts suitable for database search. It is for example common that user's give vague temporal descriptions that has to be mapped to more precise time intervals before the information can be used to access an application system.

If a request is fully specified it can be used to retrieve the desired information from the background system. There are, however, several problems related to this. For example, in cases where the background system is distributed and consists of several domain and application system knowledge sources the dialogue system must know which of them to access, in what order, and how the results should be integrated into one answer. This type of knowledge can be represented in a domain task model.

The processing of a request performed by the Domain Knowledge Manager is based on a knowledge structure called recipe. A recipe is application specific and consists of a series of service calls from different agents, which are executed in order to construct an answer to the request. Domain Knowledge Management in general involves three steps. First the Domain Knowledge Manager has to decide how to treat the request, i.e. to produce one or more recipes. In most cases one recipe is enough, but sometimes the user has provided ambiguous information that cannot be resolved by the interpreter or the Dialogue Manager, in which cases several recipes are needed. The next step is to process the recipe(s). The processing must be carefully monitored and aborted if an error occurs. Finally alternatives must be inspected and integrated into one answer that can be sent back to the Dialogue Manager.

3 Conclusions and future work

Separating domain knowledge reasoning from dialogue and task knowledge reasoning has a number of advantages. First of all, it is clearer what the responsibilities and possibilities of the different modules are, e.g. the dialogue manager handles the dialogue and not domain reasoning. Furthermore, it facilitates customisation to new application domains. Another important feature is that domain knowledge sources can easily be replaced, added, removed, and reused. This implies that a system can be made more intelligent by adding new domain agents without changing the dialogue and task models.

Future challenges are to apply the proposed architecture, utilising a Domain Knowledge Manager, to other domains and types of dialogue systems, such as advisory or tutoring systems. For such systems other knowledge sources like user models and argumentation models are relevant and have to be incorporated in the system architecture.

References

Annika Flycht-Eriksson and Arne Jönsson. 2000. Dialogue and domain knowledge management in dialogue systems. In 1st SIGdial Workshop on Discourse and Dialogue, Hong Kong.