Dialogue and Domain Knowledge Management in Dialogue Systems

Annika Flycht-Eriksson and **Arne Jönsson**
Department of Computer and Information Science
Linköping University, SE-581 83, LINKÖPING, SWEDEN
annfl@ida.liu.se arnjo@ida.liu.se

Abstract

Intelligent dialogue systems must be able to respond properly to a variety of requests involving knowledge of the dialogue, the task at hand, and the domain. This requires advanced knowledge reasoning performed by various processing modules. We argue that it is important to understand the nature of the various reasoning mechanisms involved and to separate not only, for instance, interpretation, generation, and dialogue management but also domain knowledge and task reasoning. This facilitates portability of the dialogue system to new domains and makes it easier to enhance its capabilities. In this paper we will focus on the dialogue and domain knowledge reasoning components and show how they can cooperate to achieve natural interaction.

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 it differs from the information retrieval problem discussed for instance in Stein et al. (1999). We assume that the tasks are well-defined and that the users have articulated information needs that they can express in specific terms.

In this paper we will discuss how these different tasks can be performed in dialogue systems of simple service character, i.e. dialogue systems that can provide information given a set of parameters collected from the user (Hayes and Reddy, 1983).

2 Types of requests and clarifications

Users interacting with a dialogue system utilise various communicative acts. Bunt (1989) makes a distinction between factual information acts and dialogue control acts. The latter is used to control the dialogue and the former involves any transfer of factual information. Factual information requests can be further divided into two basic types of requests:

- Task related requests. Requests where the response from the dialogue system includes domain and task specific information
- System related requests. Requests where the response includes information on what can be done with the system or pointers to other information sources

To be able to respond to questions on the system's capabilities and how to interpret the provided information, the dialogue system needs to represent knowledge about itself, here called system information. Also, if an answer can not be found in the application system(s) the dialogue system should give as helpful information as possible, for example suggesting other resources the user can consult. For this purpose knowledge is needed on where such information can be found.

The requests for task related information can be divided into simple and complex requests. Simple requests are basically requests for information

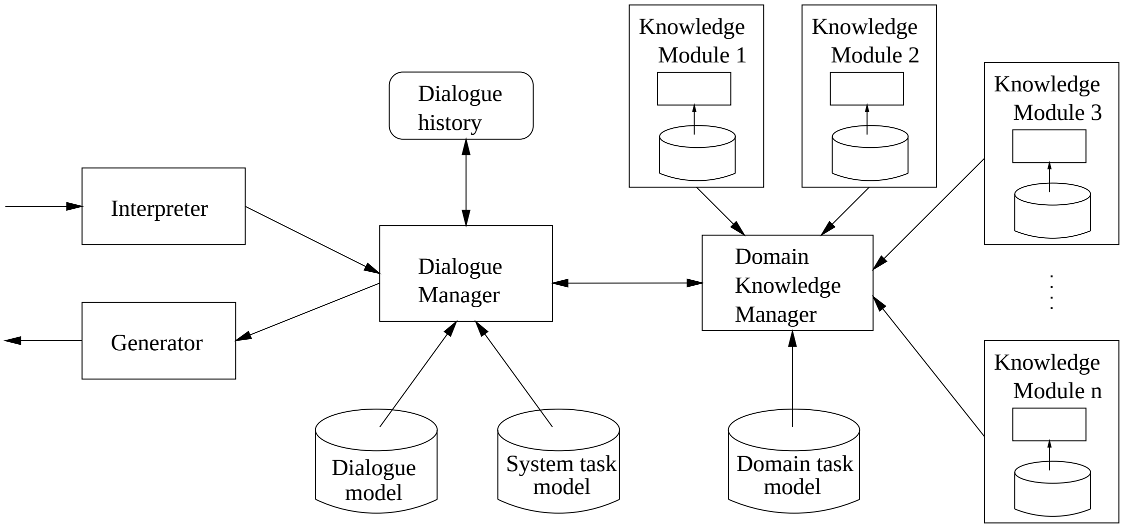

Figure 1: A dialogue system architecture. The picture shows different processing modules: Interpreter, Generator, Dialogue Manager and Domain Knowledge Manager. Some of the knowledge sources: dialogue model, domain task model, system task model, and various knowledge modules, are also depicted, but not the grammar and lexicon.

concerning properties of and relations between simple objects, for which the answers can be values of properties or names of objects. A simple object is typically an entity that can be identified by a name or a set of distinguishing features. Simple requests can be specified by an arbitrary set of parameters. The parameters describe certain properties which constraints the search for an object, or the requested properties of an object or set of objects. A typical example of a simple request is *How fast is a Volvo 850?*, which can be directly mapped onto a structure specifying that the requested object is 'Volvo 850' and the property requested is its 'speed', which in turn can be converted to an application system request.

Complex requests on the other hand are concerned with the specification and construction of compound objects. The specification of such an object requires that the user provides information on a specific set of parameters, which often involves several dialogue turns. The specification is used to construct a matching object by retrieving, and sometimes integrating, knowledge from one or several domain and application knowledge sources. Examples of complex requests are found in timetable information applications, such as the ATIS dialogues. To answer requests on a trip, the system needs to have a number of parameters specified, such as departure and arrival time and place, before it is able to access the timetables. However, for such systems there are also simple requests that can be directly mapped to a request from the background system, for instance, requests regarding meals on a flight that can be identified by a flight number, e.g. *Is breakfast served on flight SK2818?*.

Since requests are specified by a set of entities the system needs capabilities to identify entities from descriptions (Hayes and Reddy, 1983). An attempt to map a description to an entity can have three different outcomes, a unique entity is found, the description is ambiguous and corresponds to several objects, or the description is unsatisfiable and no matching object can be found. There exist several strategies to deal with these problems, but all of them include some clarification from the user or domain reasoning. In dealing with ambiguous descriptions the system should be able to provide options or find a distinguishing feature that can be used to ask the user for clarification. Unsatisfiable descriptions can be dealt with in three different ways: inform the user of the problem giving as helpful information as possible, find near misses by relaxing some of the features in the description, or find and inform the user of faulty presuppositions.

3 Dialogue system architectures

Dialogue systems often have a modular architecture with processing modules for interpretation, dialogue management, background system access, and generation, see figure 1. The processing modules utilise a number of knowledge sources, such as, grammar, lexicon, dialogue model, domain model, and task model (for an overview of some systems, see Flycht-Eriksson (1999)). In this paper focus is on dialogue management and domain knowledge management, which includes background system access.

3.1 Dialogue management

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. There are various proposals on dialogue models which can be divided in two groups: intention-based and structurally based. They differ in how they model the dialogue, especially if the user's goals and intentions behind the utterance need to be captured or not. Structurally based models are often controlled using a dialogue grammar whereas intention-based utilise plan operators. Furthermore, plan-based systems use plan operators to model not only dialogue knowledge but also task, domain and meta knowledge (c.f. Lambert and Carberry (1991), Ramshaw (1991), Ferguson et al. (1996)). This allows for plan recognition to be the only processing mechanism needed.

The *System Task model* represents how the system's tasks are performed, cf. Application Description (Hagen, 1999). 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) (van Loo and Bego, 1993; Dahlbäck and Jönsson, 1999). 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 (cf. Wahlster and Kobsa (1989)). 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 time-table 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 (cf. Bennacef et al. (1996)). It is used by the Dialogue Manager when collecting user information in order to perform a background system access. In plan-based systems the domain models takes a similar role, but wider as they often also involves advanced problem solving. We will in this paper not consider user task models, only system task models.

The *Dialogue history* records the focus of attention (Grosz and Sidner, 1986) and contains information about objects, properties, and relations as well as other dialogue information such as speech act information and system task information.

3.2 Domain Knowledge Management

If a request is fully specified it can be used to retrieve the desired information from a background system. This task is seldom discussed in literature on dialogue systems, perhaps because it is considered a rather straight forward task. 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*.

Other problems related to domain knowledge reasoning and application access where mentioned in section 2. 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 *domain knowledge sources*, called knowledge modules in figure 1. 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

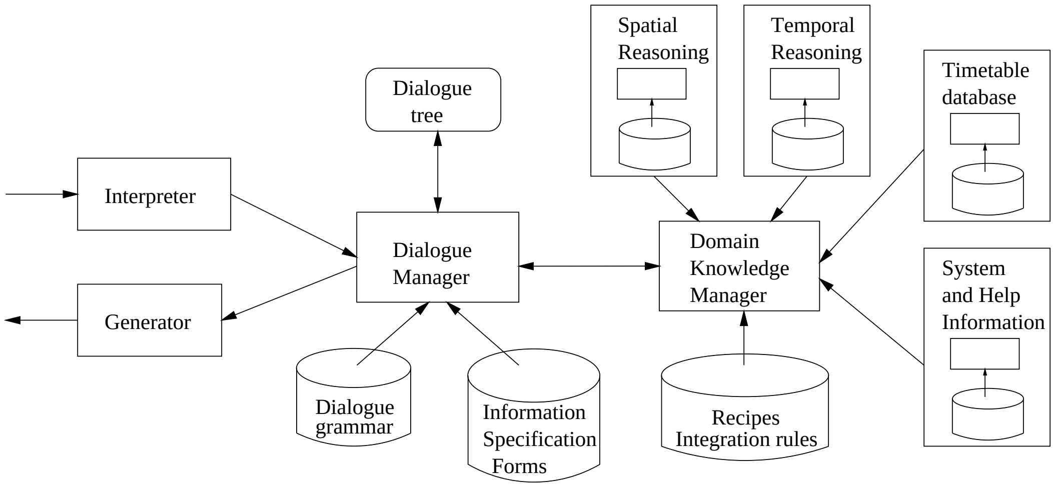

Figure 2: The MALIN dialogue system architecture in an application for local bus traffic time-table information. The dialogue model used is a dialogue grammar, the dialogue history is modelled as a dialogue tree, and Information Specification Forms correspond to the system task model. The domain and application knowledge modules perform spatial and temporal reasoning, and provide time-table and system information controlled by recipes and integration rules.

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.

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.

4 MALIN

In what follows we describe and exemplify a dialogue system with separate modules for dialogue management and domain knowledge management. The presentation will be based on the MALIN dialogue system architecture[1], figure 2, which has been used to implement an application for time-table information for local bus traffic in Östergötland.

One issue in the design of a dialogue system is how to control the various modules and the user interaction. In some systems there is no module responsible for the communication, instead a separate module, called hub (Aberdeen et al., 1999) or facilitator (Martin et al., 1999), is used for coordinating the modules and the internal information flow. Alternatively, the Dialogue Manager is the central unit of the system where the overall system behaviour is determined.

The approach taken in MALIN is a combination where a Dialogue Manager is the central controller of the interaction and the Domain Knowledge Manager is based on an agent architecture.

[1] MALIN (Multi-modal Application of LINLIN) is a refinement of the LINLINsystem (Ahrenberg et al., 1990; Jönsson, 1997) to handle also multi-modal interaction and more advanced applications.

4.1 The Dialogue Manager

In the MALIN dialogue model the dialogue is structured in terms of discourse segments, and a discourse segment in terms of moves and embedded segments. Utterances are analysed as linguistic objects which function as vehicles for atomic move segments. An initiative-response (IR) structure determines the compound discourse segments, where an initiative opens the IR-segment by introducing a new goal and the response closes the IR-segment (Dahlbäck, 1991). The discourse segments are classified by general speech act categories, such as question (Q) and answer (A) (Jönsson, 1997), rather than specialised (cf. (Hagen, 1999)), or domain related (Alexandersson and Reithinger, 1995). The action to carry out for the Dialogue Manager, as modeled in a dialogue grammar, depends on how domain entities are specified and their relation to other entities in the domain and the dialogue history.

In the MALIN dialogue system architecture there is only one dialogue history maintained by the Dialogue Manager. Thus, the other modules in the system have no memory of the previous interaction since this could cause conflicts. The dialogue history records focal information, that is, what has been talked about and what is being talked about at the moment. It is used for dialogue control, disambiguation of context dependent utterances, and context sensitive interpretation. The dialogue history is represented as a dialogue tree. The nodes in the dialogue tree record information utilising various information structures depending on the application.

For simple information requests we have identified two important concepts, termed Objects and Properties (Jönsson, 1997) where Objects models the set of objects in the database and Properties denotes a complex predicate ascribed to this set. The parameters Objects and Properties are application dependent. We also utilise Markers for various purposes (Jönsson and Strömbäck, 1998), but they will not be further discussed in this paper. Structures that represent information about objects and properties (and markers) are termed OPMs. Figure 3 shows an example OPM which represents the request *Which bus lines passes the North gate?*.

$$\begin{bmatrix} Obj: & \begin{array}{l} \#1 \begin{bmatrix} Busline: & ? \end{bmatrix} \\ \#2 \begin{bmatrix} Stop: & North\ gate \end{bmatrix} \end{array} \\ Prop: & \begin{bmatrix} PassesBy: & \begin{bmatrix} Line: & \#1 \\ Stop: & \#2 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 3: An OPM for the utterance *Which bus lines passes the North gate?*.

For complex requests the Dialogue Manager needs an information structure that holds the parameters needed before successful access of the background system can be performed. We call such structures Information Specification Forms (ISFs) (Dahlbäck and Jönsson, 1999). Just like OPMs the ISFs are application dependent and besides holding information they are also used as system task models, i.e. to inform the Dialogue Manager which parameters that has to be provided by the user. We have identified a number of different user information needs (Qvarfordt, 1998) for which ISFs are needed. The most common, called trip information, occurs when the user needs to know how and when on a particular day, most often the present day, one can travel from one point to another in town by bus. An ISF for such requests model information on departure and arrival destinations and information on arrival and/or departure time, which is required information. The user can also give information about the travel type, but this is optional. Figure 4 shows an empty Trip ISF.

$$\begin{bmatrix} Type: & Trip \\ Arr: & req. \\ Dep: & req. \\ TTime: & req. \\ TType: & opt. \end{bmatrix}$$

Figure 4: An empty trip ISF.

Another common information need, called route information, is when the caller wants information on which busses or trains that go from one point to another. This ISF is similar to the Trip ISF but time information is no longer required.

For the time-table information application both structures, ISF and OPM, are needed. This is not the case for all types of applications but we believe that if an ISF is needed an OPM can also often be useful.

4.2 The Domain Knowledge Manager

The domain knowledge sources and application systems in MALIN are implemented as agents and will from now on be called domain agents. Domain agents provide different *services*, typically to retrieve and reason about some information given some parameters, and can also request services from each other. Communication and cooperation among the agents are achieved by passing messages.

Agent	*Service*
Spatial Reasoning Agent	getBusStops(From.BusStop, From.Place, From.Street, From.Area, From.Town, FromBusStops)
Spatial Reasoning Agent	getBusStops(To.BusStop, To.Place, To.Street, To.Area, To.Town, ToBusStops)
Temporal Reasoning Agent	getTime(TTime.Time, TravelTime)
Timetable Agent	getTrips(FromBusStops, ToBusStops, TravelTime)

Figure 5: An example of an uninstantiated recipe for trip information.

U1: I want to go to the city center.
S2: The city center is a big area. Can you point in the map or give more specific information like a landmark or a street?
U3: Are there any bus stops near the Garden square?
S4: There are several bus stops near the Garden square.
< *Shows the bus stops in the map* >
U5: Then I want to go there from the University.
S6: When do you want to go?
U7: On the 31st of April before lunch.
S8: The 31st is not a valid date, there are only 30 days in April. Give a new date please.
U9: The 30th of April.
S10: The alternative trips are shown in the table.
< *Shows a table of trips* >

Figure 6: A hypothetical dialogue with the MALIN dialogue system for a local bus time-table information application. The dialogue is constructed based on a corpus of 43 dialogues collected with users of the current information service in order to illustrate some of the features of the dialogue and domain knowledge managers and our multi-modal system.

In the application of MALIN to time-table information, four different domain agents are used, see figure 2. The Temporal Reasoning Agent contains a calendar and reasons about temporal expressions. The Spatial Reasoning Agent utilises a Geographical Information System and reasoning mechanism used to deduce the relations between geographical objects (Flycht-Eriksson and Jönsson, 1998). The Timetable Agent retrieves time-table information for local bus and train traffic from an Internet source. There is also a System Information Agent which provides system information like references to human operators for questions outside the scope of time-table information.

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 a specific request, see figure 5 for an example. 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. For more details on the Domain Knowledge Manager, see Flycht-Eriksson (2000).

4.3 Communication between DM and DKM

To illustrate how the Dialogue Manager (DM) and the Domain Knowledge Manager (DKM) cooperates in processing of requests and handling of clarifications, consider the hypothetical dialogue shown in figure 6. The dialogue tree in figure 7 shows the resulting structure of the dialogue.

The first utterance, U1, initiates a trip ISF. Information about the arrival location provided by the user is inserted in the ISF in the field Arr,

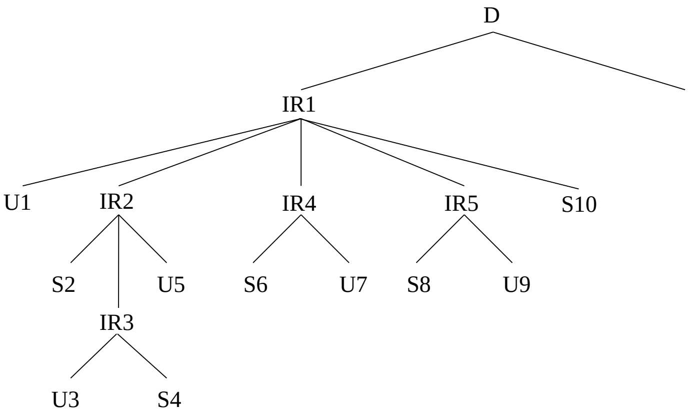

Figure 7: The dialogue tree resulting from the dialogue in figure 6.

which results in the structure presented in figure 8 included in IR1 in the dialogue tree. The ISF indicates that information about the departure place and time has to be further specified by the user by the marker *req* in the fields Dep and TTime (TravelTime).

$$\begin{bmatrix} Type: & Trip \\ Arr: & \begin{bmatrix} Area: & City\ center \end{bmatrix} \\ Dep: & req. \\ TTime: & req. \\ TType: & opt. \end{bmatrix}$$

Figure 8: The ISF in IR1 after processing of U1.

However, before continuing the dialogue and asking the user for the information that is missing in the ISF, the DM asks the DKM to validate the provided values. This validation is performed in order to detect vague or erroneous information that might have been given by the user.

The arrival location in a trip ISF will be used to find suitable bus stops that can be used to search the time-table database. The validation of the arrival location therefore means that the Spatial Reasoning Agent tries to map the location to a small set of bus stops. In this case it discovers that *Area: City Centre* is a too vague description since it corresponds to too many stops, in our case more than 5 stops. The DM is informed of this and is also given the information that more specific information like a point, a landmark or a street is required, figure 9. Thus, the user will not be asked to provide the value of another parameter since it would be an implicit confirmation that the arrival place is correct, instead a new IR-unit, IR2 in the dialogue tree, is created and a clarification, S2, is initiated based on the information from the DKM that indicates the problematic item, the type of problem, and a possible solution to the problem.

$$\begin{bmatrix} Status: & Error \\ Item: & \begin{bmatrix} Area: & City\ center \end{bmatrix} \\ Type: & \begin{bmatrix} TooMany: & BusStops \\ & \begin{bmatrix} Up: & 5 \end{bmatrix} \end{bmatrix} \\ Solution: & \begin{bmatrix} SpecInfo: & \{Point, \\ & Landmark, \\ & Street\} \end{bmatrix} \end{bmatrix}$$

Figure 9: The response from the DKM to the domain validation of the arrival location.

Instead of answering the system's question the user takes the initiative by requesting new information, U3. This request results in a new IR-unit, IR3, to be inserted in the dialogue tree as a clarification of the system's clarification in IR2, as shown in figure 7. The utterance is a simple request and the DM utilises an OPM to model this, figure 10.

$$\begin{bmatrix} Obj: & \begin{bmatrix} \#1 \begin{bmatrix} Stop: & ? \end{bmatrix} \\ \#2 \begin{bmatrix} Landmark: & Garden \\ & square \end{bmatrix} \end{bmatrix} \\ Prop: & \begin{bmatrix} Near: & \begin{bmatrix} Place1: & \#1 \\ Place2: & \#2 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 10: The OPM in IR3 after processing of U3.

To answer this request means reasoning about spatial relations between geographical objects. The request is therefore sent to the DKM which asks the Spatial Reasoning Agent for information. The request is successfully processed and some nearby bus stops are found and sent back to the DM utilising the structure in figure 11. The DM can then ask the generator to present them to the user, S4.

$$\begin{bmatrix} Status: & Success \\ Stops: & \begin{bmatrix} \begin{bmatrix} Name: & Centrum \\ & Snickareg.\ 30 \\ Id: & 1268 \end{bmatrix} \\ \begin{bmatrix} Name: & Linnegatan \\ Id: & 1220 \end{bmatrix} \\ \begin{bmatrix} Name: & Stora\ torget \\ Id: & 450 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 11: The response from the DKM to the OPM in IR3.

The user responds to this answer by confirming his departure location, U5, and thereby responds to the request S2 of IR2. He also provides an arrival location. This new information is represented in the OPM of IR2, figure 12.

$$\begin{bmatrix} Obj: & \begin{bmatrix} \#1 \begin{bmatrix} Landmark: & Garden \\ & square \end{bmatrix} \\ \#2 \begin{bmatrix} Landmark: & University \end{bmatrix} \end{bmatrix} \\ Prop: & \begin{bmatrix} Arr: & \#1 \\ Dep: & \#2 \end{bmatrix} \end{bmatrix}$$

Figure 12: The OPM in IR2 after processing of U5.

The DM resumes processing of the ISF in IR1 and updates it with the arrival and departure location based on the information in the OPM of IR2. Information about the arrival location is added to the previously provided information in the field Arr. The new information about the departure location is inserted in the field Dep, yielding the structure in figure 13.

$$\begin{bmatrix} Type: & Trip \\ Arr: & \begin{bmatrix} Area: & City\ center \\ Landmark: & Garden\ square \end{bmatrix} \\ Dep: & \begin{bmatrix} Landmark: & University \end{bmatrix} \\ TTime: & req. \\ TType: & opt. \end{bmatrix}$$

Figure 13: The ISF in IR1 after updates with information from the subtree in IR2.

Again the DM asks the DKM for domain validation of the partially specified ISF. Since both locations can be mapped to a limited number of bus stops the ISF is approved by the DKM. The DM now needs to have a time to complete the ISF, and consequently a new IR-unit, IR4 in the dialogue tree, is created and the user is, in utterance S6, asked for this. The answer U7 is a valid response to S6 and produces a new OPM, see figure 14.

$$\begin{bmatrix} Obj: & \#1 \begin{bmatrix} Date: & \begin{bmatrix} Day: & 31 \\ Month: & April \end{bmatrix} \\ Time: & \begin{bmatrix} POD: & lunch \\ Mod: & before \end{bmatrix} \end{bmatrix} \\ Prop: & \begin{bmatrix} TTime: & \#1 \end{bmatrix} \end{bmatrix}$$

Figure 14: The OPM in IR4 after processing of U7.

The new information from IR4 is then inserted as TTime in the ISF of IR1. This results in a fully specified Trip ISF, figure 15.

$$\begin{bmatrix} Type: & Trip \\ Arr: & \begin{bmatrix} Area: & City\ center \\ Landmark: & Garden\ square \end{bmatrix} \\ Dep: & \begin{bmatrix} Landmark: & University \end{bmatrix} \\ TTime: & \begin{bmatrix} Date: & \begin{bmatrix} Day: & 31 \\ Month: & April \end{bmatrix} \\ Time: & \begin{bmatrix} POD: & lunch \\ Mod: & before \end{bmatrix} \end{bmatrix} \\ TType: & opt. \end{bmatrix}$$

Figure 15: The ISF of IR1 after updates with information from IR4.

The ISF is again sent to the DKM for validation. When the Temporal Reasoning Agent tries to map the temporal description in TTime to a format suitable for time-table database search it discovers the erroneous date. The DKM then returns a response, figure 16, to the DM informing it of the error. The DM initiates a new clarification IR-unit, IR5, and a clarification is formulated, S8.

$$\begin{bmatrix} Status: & Error \\ Item: & \begin{bmatrix} Date: & \begin{bmatrix} Day: & 31 \\ Month: & April \end{bmatrix} \end{bmatrix} \\ Type: & \begin{bmatrix} NotValid: & \begin{bmatrix} Month: & April \\ Up: & 30 \end{bmatrix} \end{bmatrix} \\ Solution: & \begin{bmatrix} SpecInfo: & \{Date\} \end{bmatrix} \end{bmatrix}$$

Figure 16: The response from the DKM to the domain validation of the time description.

The user responds to the system's clarification request and provides a new date, U9. The response is modelled in an OPM in IR5, figure 17.

$$\begin{bmatrix} Obj: & \#1 \begin{bmatrix} Date: & \begin{bmatrix} Day: & 30 \\ Month: & April \end{bmatrix} \end{bmatrix} \\ Prop: & \begin{bmatrix} TTime: & \#1 \end{bmatrix} \end{bmatrix}$$

Figure 17: The OPM of IR5 after U9.

The information in the clarification request IR-unit, IR5, is propagated to the ISF of IR1 which is updated. This time the new information replaces the old in TTime since it was erroneous. The resulting ISF is presented in figure 18.

$$\begin{bmatrix} Type: & Trip \\ Arr: & \begin{bmatrix} Area: & Citycenter \\ Landmark: & Gardensquare \end{bmatrix} \\ Dep: & \begin{bmatrix} Landmark: & University \end{bmatrix} \\ TTime: & \begin{bmatrix} Date: \begin{bmatrix} Day: & 30 \\ Month: & April \end{bmatrix} \end{bmatrix} \\ Time: & \begin{bmatrix} POD: & lunch \\ Mod: & before \end{bmatrix} \\ TType: & opt. \end{bmatrix}$$

Figure 18: The ISF of IR1 after integration with the information in IR5.

Once more a validation of the ISF is performed by the DKM. This time no problems are detected and a search for suitable trips can finally be done. The DKM does this by first asking the Spatial Reasoning Agent to map the departure and arrival locations to two sets of bus stops, then asking the Temporal Reasoning Agent to map the vague temporal description to a precise time interval. Given this information the DKM then searches the timetable database to find one or more trips that fulfill the requirements. The resulting trips are sent back to the DM and displayed to the user, S10.

4.4 Implementation

The MALIN dialogue system customised for the traffic information application is currently under development. The Dialogue Manager from the LINLIN dialogue system architecture has been adapted to allow also ISFs and we are currently specifying the dialogue grammar and how to handle focus tracking utilising ISFs and OPMs at the same time.

The Domain Knowledge Manager is functional utilising a Spatial Reasoner for one sub-area of Östergötland and a Temporal Reasoner. The Timetable Agent retrieves trip information from the current Internet based timetables. Recipes are developed for accessing these modules, but the System and Help Information knowledge source is not yet implemented.

5 Conclusions and future work

In this paper we have presented an architecture for dialogue systems where a Domain Knowledge Manager and a Dialogue Manager cooperate to achieve natural interaction. Information providing dialogue systems based on this architecture can handle a variety of requests; simple and complex concerning the domain, and requests for system related information.

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.

6 Acknowledgments

This work is supported by The Swedish Transport & Communications Research Board (KFB) and the Center for Industrial Information Technology (CENIIT). We are indebted to Lars Degerstedt, Håkan Johansson and Lena Santamarta for fruitful discussions.

References

John Aberdeen, Sam Bayer, Sasha Caskey, Lauire Damianos, Alan Goldschen, Lynette Hirschman, Dan Loehr, and Hugo Trappe. 1999. Implementing practical dialogue systems with the DARPA communicator architecture. In *Proceedings of IJCAI'99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems, August, Stockholm.*

Lars Ahrenberg, Arne Jönsson, and Nils Dahlbäck. 1990. Discourse representation and discourse management for natural language interfaces. In *Proceedings of the Second Nordic Conference on Text Comprehension in Man and Machine, Täby, Sweden.*

Jan Alexandersson and Norbert Reithinger. 1995. Designing the dialogue component in a speech translation system. In *Proceedings of the Ninth Twente Workshop on Language Technology (TWLT-9)*, pages 35–43.

S. Bennacef, L. Devillers, S. Rosset, and L. Lamel. 1996. Dialog in the RAILTEL telephone-based system. In *Proceedings of International Conference on Spoken Language Processing, ICSLP'96*, volume 1, pages 550–553, Philadelphia, USA, October.

Harry C. Bunt. 1989. Information dialogues as communicative action in relation to partner modelling and information processing. In M. M. Taylor, F. Néel, and D. G. Bouwhuis, editors, *The Structure of Multimodal Dialogue*, pages 47–73. Elsevier Science Publishers B.V. (North-Holland).

Nils Dahlbäck and Arne Jönsson. 1999. Knowledge sources in spoken dialogue systems. In *Proceedings of Eurospeech'99*, Budapest, Hungary.

Nils Dahlbäck. 1991. *Representations of Discourse, Cognitive and Computational Aspects.* Ph.D. thesis, Linköping University.

George Ferguson, James Allen, and Brad Miller. 1996. TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third Conference on Artificial Intelligence Planning Systems, AIPS-96*, pages 70–77.

Annika Flycht-Eriksson and Arne Jönsson. 1998. A spoken dialogue system utilizing spatial information. In *Proceedings of International Conference on Spoken Language Processing, ICSLP'98*, page 1207, Sydney, Australia.

Annika Flycht-Eriksson. 1999. A survey of knowledge sources in dialogue systems. In *Proceedings of IJCAI'99 workshop on Knowledge and Reasoning in Practical Dialogue Systems, August, Stockholm*, pages 41–48.

Annika Flycht-Eriksson. 2000. A domain knowledge manager for dialogue systems. In *Proceedings of the 14th European Conference on Artificial Intelligence, ECAI 2000.* IOS Press, Amsterdam.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intention and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Eli Hagen. 1999. An approach to mixed initiative spoken information retrieval dialogue. *User modeling and User-Adapted Interaction*, 9(1-2):167–213.

Philip J. Hayes and D. Raj Reddy. 1983. Steps toward graceful interaction in spoken and written man-machine communication. *International Journal of Man-Machine Studies*, 19:231–284.

Arne Jönsson and Lena Strömbäck. 1998. Robust interaction through partial interpretation and dialogue management. In *Proceedings of Coling/ACL'98, Montréal.*

Arne Jönsson. 1997. A model for habitable and efficient dialogue management for natural language interaction. *Natural Language Engineering*, 3(2/3):103–122.

Lynn Lambert and Sandra Carberry. 1991. A tripartite plan-based model of dialogue. In *Proceedings of the 29th Annual Meeting of the ACL, Berkeley*, pages 193–200.

David L. Martin, Adam J. Cheyer, and Douglas B. Moran. 1999. The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1-2):91–128, January-March.

Pernilla Qvarfordt. 1998. Usability of multimodal timetables: Effects of different levels of domain knowledge on usability. Master's thesis, Linköping University.

Lance A. Ramshaw. 1991. A three-level model for plan exploration. In *Proceedings of the 29th Annual Meeting of the ACL, Berkeley*, pages 39–46.

Adelheit Stein, Jon Atle Gulla, and Ulrich Thiel. 1999. User-tailored planning of mixed initiative information-seeking dialogues. *User Modeling and User-Adapted Interaction*, (9):133–166.

Wim van Loo and Harry Bego. 1993. Agent tasks and dialogue management. In *Workshop on Pragmatics in Dialogue, The XIV:th Scandinavian Conference of Linguistics and the VIII:th Conference of Nordic and General Linguistics*, Göteborg, Sweden.

Wolfgang Wahlster and Alfred Kobsa. 1989. User models in dialog systems. In *User Models in Dialog Systems.* Springer-Verlag.