Authors: R. Catizone, A. Setzer, Y. Wilks Date: September 2002

Deliverable D5.1 State of the Art in Dialogue Management

Document History

Version	Editor	Date	Explanation	Status
0.1	Catizone, Setzer, Wilks	August 2002	Draft version to be read by the partners.	Draft Version
0.2	Catizone, Setzer, Wilks	September 2002	Incorporated comments from other partners.	Review
1.0	Catizone, Setzer, Wilks	September 2002	PCC Approval	Final

COMIC

Information sheet issued with Deliverable D5.1

Title:	State of the Art in Dialogue Management
Abstract:	In this document, the state of the art in dialogue management is presented. This includes multi-modal and uni-modal approaches. As well as giving a general overview of the types of approaches one can take towards dialogue and action management, existing systems (either current or of particular interest) are described as well. After having presented the state of the art, conclusions are drawn and recommendations for the COMIC projects given.
Author(s):	R. Catizone, A. Setzer, Y. Wilks
Reviewers:	
Project:	COMIC
Project number:	IST- 2001-32311
Date:	September 2002
	For Public Use
Key Words:	Dialogue, multi-modal, uni-modal, systems,
Distribution List:	
COMIC partners	Max Planck Institute for Psycholinguistics, Deutsches Forszungszentrum für Künstliche Intelligenz GmbH (DFKI), Max Planck Institute for Biological Cybernetics, University of Sheffield, University of Edinburgh, ViSoft, Katholieke Universiteit Nijmegen.
External COMIC	Public

Contents

1 Introduction

This report reviews the state of the art research in Dialogue Management Systems in 2002. We have tried to focus on multimodal systems where possible but, given that most research in Dialogue Management has been uni-modal, much of our survey concerns Dialogue Management systems in a uni-modal context keeping in mind a multi-modal extension. Dialogue systems have been around since the 60's of which the best known are conversation programs such as Eliza (Wiezenbaum 1966) and Parry (Colby 1973). Since that time many approaches to dialogue management systems have been implemented and many surveys on the topic have been produced. In writing this document, we have reviewed some of the more recent dialogue management systems (totalling 11) that we felt were technologically significant or had a particular closeness to the problem that we are researching in the COMIC project. We have not looked closely at older systems, since they have already been discussed in other state-of-the-art dialogue management system surveys. However we have made reference to/included in our discussion many of those systems as described in other reports. One of the key survey papers in dialogue management is that of Churcher, Atwell and Souter (Churcher 1997)). We feel that the categories they use for the various approaches to dialogue management are still descriptive of the systems in the field. The approaches we describe in section 2 are as follows: finite state/dialogue grammars, plan-based and collaborative. However, like all categorisation schemes, not all systems fall neatly into one category or the other. It is also worth noting that, given the years of research behind dialogue management, new approaches are scarce.

We did notice that there seemed to be an inconsistency regarding the types of approach. Finite state models were contrasted with plan-based and collaborative approaches. This seems to be dividing systems inconsistently – as design approaches and at the same time implementation approaches. Plan-based vs. collaborative seems a reasonable distinction about design, but to contrast with finite state implementations seems wrong. Finite state models can be used to implement a variety of approaches independently of the design choice. Again, collaborative models may or may not be plan-based, so this distinction too, is less than firm.

The COMIC project demonstrator consists of a multimodal graphical system using speech and pen input for helping the user to design a bathroom. The spoken output will be delivered through a bathroom design consultant in the form of a talking head avatar with natural facial expressions. The type of multimodal conversation that COMIC will handle will be to suggest various kinds of bathroom furnishings to the user as well as offering aesthetic judgements of bathroom design. In our model's role as bathroom design advisor, we will have to take into consideration logical inconsistencies when they arise to prevent the user from producing a design that is incoherent in practice.

This document consists of 4 main parts; an introduction, a description of the broad categories of Dialogue Management approaches, a detailed description of the particular dialogue management systems that we surveyed grouped by application functionality, and finally the conclusion section where we offer our recommendation as to how the dialogue management system in COMIC should be designed.

2 Approaches

In this section, we will give a brief summary of how Dialogue Management systems have traditionally been classified, based on Churcher (1997) and Cohen (1995).

A system involving dialogue management will normally consist of the following components (see Figure 1):

multi- or uni-modal input,

multi- or uni-modal output,

the application itself,

meaning extraction,

response generation, and

the dialogue manager (DM).

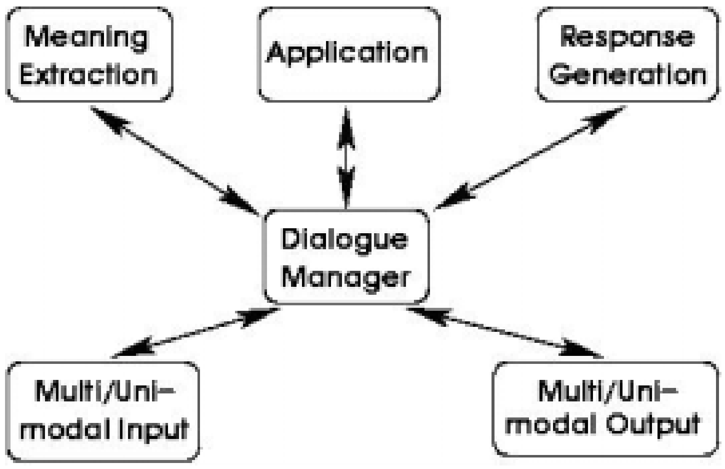

Figure 1: Typical DM scenario

Multi/uni-modal input is analysed (e.g. text or speech input is parsed and a semantic representation is assigned by the appropriate components. The DM then infers the implications of the input within the current context, resolves conflicts, ambiguities, and deals with anaphora and ellipsis. The response generator generates a response based on the output from the DM, and decides which modality to use.

Traditionally, approaches to Dialogue Management have been classified into the following three categories: dialogue grammars, plan-based approaches and collaborative approaches. The approaches are not mutually exclusive, and are often used together. Also, plan-based approaches include features of dialogue grammars, and collaborative approaches include features of plan-based approaches. We will give a brief description of each of them.

2.1 Dialogue Grammars

Dialogue grammars, in our definition for this survey, are systems that identify and represent local or global surface patterns of dialogue or patterns of speech acts (Searle 1969) and responses.

Dialogue grammars, which have a long history (Polany and Scha, 1984; Reichman, 1981; Sinclair and Coulthard, 1975), use prescriptive grammars for sequences in dialogues. The first grammars described the structure of the complete dialogue, from beginning to end, whereas more recent approaches are based on the observation that there are a number of sequencing regularities in dialogues, which are called **adjacency pairs**.

It has been proposed that a dialogue is a collection of such pairs (Jefferson, 1972), which describe facts such as that questions are generally followed by answers, proposals by acceptances, denials etc. Digressions and repairs are dealt with by using embedded sequences.

Dialogue grammars are used to parse the structure of a dialogue, just as syntactic grammar rules are used to parse sentences. Phrase-structure grammar rules and various kinds of state machines have been used to implement dialogue grammars. For example the SUNDIAL system, described in section 3.1.1, uses a dialogue grammar to engage in dialogue about travel conversations.

Although dialogue grammars have been successfully implemented (Müller and Runger, 1993; Nielsen and Baekgaard, 1992), they have been criticised on the grounds that they lack flexibility both to deviations in the dialogue as well as applicability to other domains.

Another extension of dialogue grammars are **frame-based approaches**, which have been developed to overcome the lack of flexibility of dialogue grammars. The entities in the application domain are hierarchically modelled, and the system can control the dialogue according to the requirements of those entities. Hulstijn et al. (1996), for example, who developed a theatre booking system, arranged frames hierarchically to reflect the dependence of certain topics (like the details of the performance the user wants to see) on others. In Veldhuijzen van Zanten (1996), a train timetable enquiry system, a frame structure relates the entities in the domain to one another, and this structure captures the meaning of all possible queries the user can make.

2.2 Plan-based

Plan-based approaches are based on the view that humans communicate to achieve goals, including changes to the mental state of the listener. Utterances are seen not just as strings but as performing speech acts (Searle, 1969) and are used to achieve these goals. The listener has to identify the speaker's underlying plan and respond accordingly. For example, in response to a customer's question of "Where are the steaks you advertised?", a butcher's reply of "How many do you want?" is appropriate, because the butcher understands the customer's underlying plan to buy the steaks (taken from Cohen (1995)).

Plan-based theories of communicative action and dialogue (for example: Allen and Perault, 1980; Appelt, 1985; Cohen and Levesque, 1990) claim that the speaker's speech act is part of a plan and that it is the listener's job to identify and respond appropriately to this plan. Plan-based approaches attempt to model this claim and explicitly represent the (global) goals of the task.

Plan-based approaches have been criticised on practical and theoretical grounds. For example, the processes of plan-recognition and planning are combinatorically intractable in the worst case, and in some cases they are undecidable. Plan-based approaches also lack a sound theoretical basis. There is often no specification of what the system should do, for example, in terms of the kinds of dialogue phenomena and properties the framework can handle or what the various constructs like plans, goals, etc are.

Conversational Games Theory (Carletta et al., 1995; Kowtko et al., 1991) uses techniques from both discourse grammars and plan-based approaches by including a goal or plan-oriented level in its structural approach. It can be used to model conversations between a human and a computer in a task-oriented dialogue (Williams, 1996).

A (task-oriented) dialogue consists of a one or more transactions, each transaction representing a subtask. A transaction comprises a number of conversational games, which in turn consist of an opening move, and (sometimes optional) end move. An example is an INSTRUCTION game which consists of three nested games: an EXPLAINING game, a QUERY-YN game, and a CHECKING game. The CHECKING game, for example, can consist of a QUERY-YN and a REPLY-Y or a REPLY-N.

The approach deals with discourse phenomena such as side sequences, clarifications etc. by allowing games to be have another game embedded with in it - a technique which allows for the modelling of the complexity of natural dialogue.

2.3 Collaborative

Collaborative approaches are based on viewing dialogues as a collaborative process. Both partners work together to achieve a mutual understanding of the dialogue. The motivations that this joint activity places on both partners motivates discourse phenomena such as confirmation and clarification - which are also evident in human to human conversations.

Collaborative approaches try to capture the motivations behind a dialogue and the mechanisms of dialogue itself, rather than concentrate on the structure of the task. The beliefs of at least two participants will be explicitly modelled. A proposed goal, which is accepted by the other partner(s), will become part of the shared belief and the partners will work cooperatively to achieve this goal.

Several such theories have been developed, some of which will be discussed below, and each approach uses a combination of techniques from plan-based and discourse grammar approaches.

In the TRAINS-93 dialogue manager, Traum's (1996) model of conversation agency extended Bratman's et al. (1988) Beliefs Desires Intentions (BDI) agent architecture. In the BDI model, actions in the world affect agent's beliefs and the agent can reason about its beliefs and thus formulate desires and intentions. Beliefs are how the agent perceives the world, desires are how the agent would like the world to be, and intentions are formulated plans of how to achieve these desires. Traum states two major problems with the BDI model. He argues that agent's perceptions not only influence its beliefs but also its desires and intentions. Also, the BDI model does not support more than one agent. Traum thus extended the BDI model by incorporating mutual beliefs, i.e. what both agents belief to be true and also let perceptions influence desires and intentions as well as beliefs.

Viewgen (Wilks and Ballim 1991b) is a representational system for modelling agents and their beliefs and goals as part of a dialogue system. It has two types of structures: those for agents that can have views of other agents and entities, and those for entities that have no points of view of their own. It is based on a virtual machine that nests these entities to any depth required for analysis by nesting either type of object inside the first type: i.e. agents can have perspectives of entities and other agents. The important notion is that nested beliefs (about beliefs and goals) are created only at need and not prestored in advance (a highly implausible notion) as in the Rochester-type systems that compute over goals and beliefs.

Apart from its procedural motivation, ViewGen was designed to ameliorate a key problem of AI that individuals can have beliefs that contradict each other, but which can only be contained in a partitioned system (like that of the structures above, called environments) so that explicit contradictions never meet and collapse the first order logic. It was also intended to contain the scope of inferences required for NLU, by constructing at need an appropriate nested environment and them attempting to limit all the inferences required to the contents of that particular environment, as opposed to the searching of some vast knowledge base of theorems. Figure 2 shows an example of nested environments and conflicting beliefs.

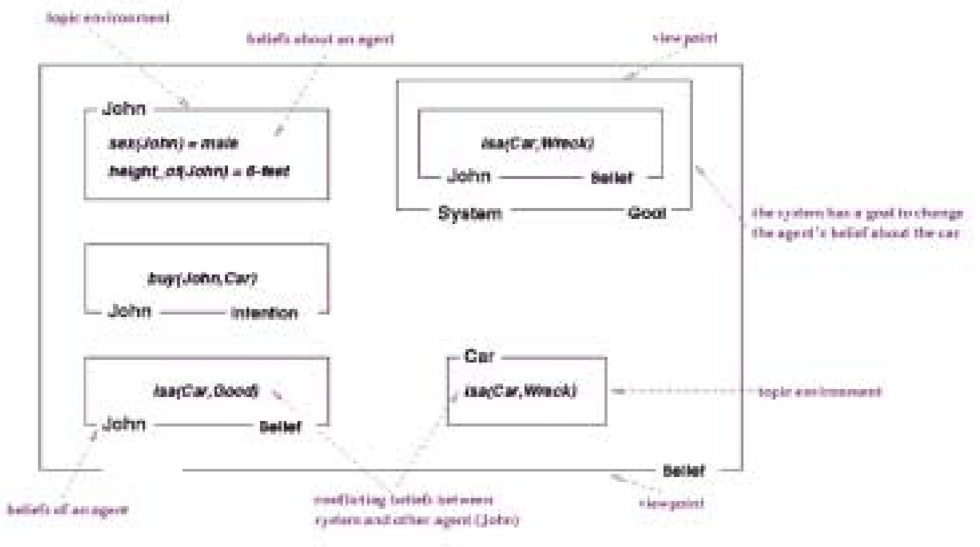

Figure 2: ViewGen example

VG was designed to be part of an overall NLP/NLU system and extended so that metaphorical sense extension could also be seen as an individual's perspective on a lexicon. The system had wider philosophical motivations: (1) to undermine notions of shared/mutual/common knowledge, showing that they are confused notions, since (in the absence of decoding heads) all belief must in the end be that of an individual, including that about other individuals or of what they mutually believe; (2) it proposed a notion of belief as such as what could be contrasted with an alternative belief in another environment (which is why bank machines should not be said to have beliefs about our accounts).

The system was initially implemented in Prolog as a dialogue system where a doctor/system could interact separately with a couple who needed genetic counselling and could be supposed top have different levels of expertise on the subject matter. It was later reimplemented (Wilks and Lee 1996) as part of an advisor on printing facilities, and embodied (Bontcheva 2001) in a system that generated hypertext appropriate to a user's level of knowledge.

Other approaches include Novick and Hansen (1995), Novick and Ward (1993), Chu-Carroll (1996), who extends Sidner's (1992, 1994), and Beun (1996).

3 Systems grouped by Function

3.1 Enquiry/Booking Systems

We are defining Enquiry Systems as those systems that provide an interface (usually spoken) to the user for acquiring information on a particular topic such as timetable information , cinema booking, banking information etc.

3.1.1 SUNDIAL

In SUNDIAL (Speech UNderstanding in DIALogue) project (McGlashan et al. 1992, Eckert 1993, McGlashan 1996), which was one of the largest speech technology projects in Europe at that time, a real-time integrated dialogue system was built. This system is capable of maintaining cooperative dialogues with users over standard telephone lines. The domain of this system, which has been developed in four languages (French, German, Italian and English), is flight reservations and enquiries, and train enquiries. Each of the four language systems carries out three principal functions: the interpretation of user utterances; the generation of system utterances; and management of the dialogue so that the system's utterances are natural and coherent. These functions are distributed over five modules. The linguistic interpretation is dealt with by an acoustic processing module and a linguistic processing module. The dialogue manager, on which we will concentrate in this overview, takes each linguistic representation and gives it an interpretation within the dialogue context. Based on this interpretation it then decides how the dialogue should be continued and, if it is the system's turn, plans a schematic linguistic representation for the system utterance. This utterance is carried out by the message generation module and the speech synthesis module.

3.1.1.1 The Dialogue Manager

The SUNDIAL project addresses two principal problems in dialogue management. First, dialogue management needs to be **generic**, i.e. it has to be able to interpret and generate utterances in more than one language and across more than one task domain. Second, dialogue management should provide a cooperative interaction with the user, which means the utterances produced should be perceived as natural, coherent, and helpful within the context of the dialogue. Speech, especially continuous speech from untrained users, causes special problems for a cooperative dialogue due to low recognition rates. Problems like these must be tackled by the dialogue manager (DM) to successfully maintain a cooperative dialogue.
The solution proposed by SUNDIAL lies in its **distributed database architecture** of the DM, a design which has both an empirical and a theoretical basis. Figure 3 shows the design. A central feature is the construction and maintenance of a model of the interaction.

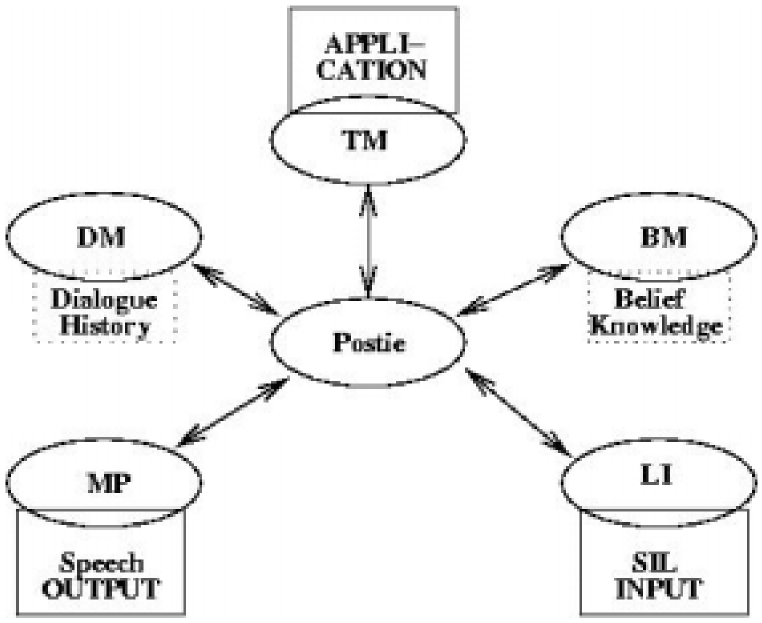

Figure 3: SUNDIAL DM architecture.

Cooperative dialogue management requires the construction and maintenance of an **interactional model**, which specifies the layers of structure which can be distinguished in dialogues. Based on Grosz and Sidner (1986) and Bunt (1989), four structural layers have been identified and are modelled in SUNDIAL: a linguistic, a belief, a dialogue, and a task structure. Each structural layer is represented in a separate sub-model, which can be seen in Figure 3.

When representing belief, the SUNDIAL approach is to capitalise on the capability of cooperating agents to evolve a common model of the discourse situation jointly. Models like this are called falsification definite, because agents assume rather than infer that their belief model corresponds to the belief model of other agents. Agents are obliged to make the state of their belief model explicit so that the other agent can correct or modify it. This approach has been chosen over a model based on plan inferences, which rely on a considerable amount of defeasible inference and which show a lack of symmetry with which the system reasons about a user's intentions but not its own.

3.1.1.2 The Distributed Database Architecture

The model of interaction that the DM builds and maintains is distributed over the five modules shown in Figure 3. Each module is composed of the relevant sub-model of the interaction and a set of rules and (apart from the message planning module) has a symmetrical representation of the system and the user. Each module is associated with a number of update rules determining how the (partial) model is extended through system and user utterances. Since these rules may depend on more than the current representation, the modules have to communicate with each other to get the relevant information and each module has a bi-directional interface to achieve this.

In the following we will briefly describe each module.

The Task Module
The task module maintains a model of the task structure of the dialogue and also consults a domain specific application database. It informs the dialogue module of the current state of the task, which involves e.g. deciding whether the user has provided enough information to perform the task.

Another important case for cooperative dialogue is when the user provides enough but incorrect information, for example when there is no flight to the stated destination city. Instead of informing the dialogue module that the task has failed, which is detected when the application database is consulted, the TM attempts to relax the task parameters and suggests alternative values. In this case, appropriate system utterances informing the user of the alternatives have to be produced and the user can clarify or modify the request.

The Belief Module
The purpose of this module is to build and maintain the common **belief model** which contains concepts created directly as a result of user utterances as well as assumed initial common ground, together with inferential extensions. The belief model is used to "derive a contextual interpretation of utterances, locating concepts corresponding to referring expressions, anchoring elliptical utterances in context, and deriving task-relevant interpretations" (McGlashan et al 1992).

Two complementary principles are used when extending the belief model. Firstly, existing structures may be visited or added to, and secondly, in the case of a potential mismatch between the knowledge of two different agents or when an agents entertains alternatives, perspectives may be used to separate out the different representations.

The belief module communicates with the dialogue module by informing it of possible contexts or anchor points as well as receiving information from the dialogue module about the resulting status of the interpretation process. In addition to that, the belief module cooperates with the message output system so that appropriate referring expressions can be planned and generated and also send information to be used for appropriate intonational accents in the speech synthesis.

The Dialogue Module
The dialogue module maintains a model of the dialogue context, builds an interpretation of user utterances, and determines how the dialogue should continue. The dialogue is seen as hierarchically structured into exchanges, interventions, and dialogue acts – based on the linguistic analysis of Roulet and Moeschler (cf. Moeschler 1989).

The dialogue module interacts with the belief module and informs it of a referential interpretation of the utterance. It also creates a set of **dialogue allowances**, a set of possible dialogue continuations given the current state of the dialogue, constructed on the basis of the effect user utterances have on the dialogue, task, and belief module. For example, if additional information is needed from the user as required from the task module, then the dialogue module may plan an allowance which requests the information from the user.

Which of the allowances is realised in the next system utterance is partly determined by the **dialogue strategy**, whose purpose it is to minimise the risk of breakdown by modulating the system output according to the progress of the dialogue. The two key factors in selecting an allowance are the acoustic scores associated with a user utterances and the degree to which exchanges are embedded.

The dialogue strategy used in SUNDIAL is fine-grained and flexible, which supports cooperative interaction with users. As long as the dialogue is running smoothly, a wide range of responses from the user is permitted by the system output. When the dialogue becomes more difficult, the system begins to control the range of user responses, e.g. by means of a `menu` style interaction.

The Linguistic Interface Module
The linguistic interface module is responsible for interfacing the dialogue module with the linguistic processing module as well as maintaining a model of the linguistic structure of the system and user turns. It also constructs predictions for the linguistic processing module based on an ordered set of dialogue allowances.

Message Planning Module
The message planning module is responsible for interfacing the dialogue module with the message generation module. It augments a dialogue allowance send from the dialogue module with rhetorical and semantic information which it receives from the belief and linguistic modules. After the utterance has been generated, the appropriate updates are sent to the dialogue, belief, and linguistic interface module.

3.1.2 SmartKom

SmartKom (Alexandersson and Becker 2001) is a multimodal dialogue system that combines, speech, gesture and mimics input and output. One of the major scientific goals of SmartKom is to design new computational methods for the seamless integration and mutual disambiguation of multimodal input and output on a semantic and pragmatic level.

3 Application Scenarios

Public - an intelligent telephone booth is developed with which one is able to book tickets, get information about different (local) activities, attractions etc.
Home
Mobile

The SmartKom Architecture

Interface modules: on the input side: there is an audio module, on the output side the display manager
Recognizers and synthesizers: on the input side, there is gesture recognition, prosody and speech recognition modules, on the output side speech synthesis and the display manager.
Semantic processing modules: this group of modules comprises or transforms meaning representations: gesture and speech analysis, media fusion, intention recognition, discourse and domain modelling, action planning, presentation planning and concept-to-speech generation.
External services: the function modelling module is the interface to external services, e.g. EPG databases, map services, and information extraction from the Web

The discourse module receives hypotheses directly from the intention analysis module. The hypotheses are validated and enriched with (consistent) information from the discourse history. During this process a score is computed which mirrors how well the hypothesis fits the history. Depending on the scores by the analysis modules and the score by the discourse modeller, the intention analysis module picks the "best" hypothesis.

Detail of the two subtasks

1. **Fill in consistent information from the history**
The current hypothesis is iteratively compared to the previous discourse segments. They use a combination of unification and overlay: discourse markers signal which parts must be overlaid, the remainder of the structures are unified. If this operation fails, the algorithm steps back one segment in the discourse history.

2. **Compute a score**
A simple heuristic is used for scoring the augmentation from discourse memory which combines the information contained in the output and the distance in discourse history.

The formal definitions are based on *typed feature structures* since unification and other operations similar to overlay are usually presented this way. All information in SmartKom is encoded in XML, in particular M3L.

3.1.3 ARISE

ARISE (**A**utomatic **R**ailway **I**nformation **S**ystem for **E**urope) is a 4th Framework project with 13 industrial and academic partners. The application of this spoken dialogue system is to provide train timetable information over the phone. Prototypes have been developed in four languages: Dutch, French, English, and Italian.

Although the four systems differ slightly, we will describe a system which is a generic description of ARISE. This description is based on Sturm et al. (1999a), Sturm et al. (1999b), Lamel et al. (1999), Lamel et al. (2000), and Baggia et al. (1999).

The ARISE system is based on a modular architecture containing six components:

1. a continuous speech recogniser,
2. a natural language understanding component,
3. a mixed-initiative dialogue manager,
4. a database retrieval component,
5. a generation component,
6. a synthesizer.

An overview of the architecture can be found in Figure 5.

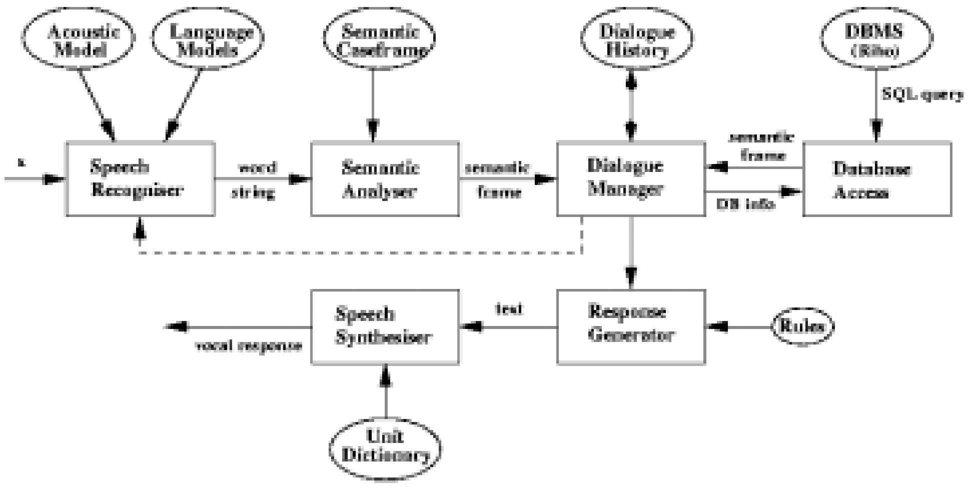

Figure 4: The LIMSI ARISE architecture

Figure 5: The LIMSI architecture.

3.1.3.1 The Dialogue Manager

A two-level dialogue strategy was employed by the Dialogue Manager (DM). A mixed-initiative approach, where the user can provide information at any time, is combined with a system-directed approach when the system detects a problem during the dialogue.

A mix of implicit and explicit confirmation is used, based on how confident the system is as to whether an item has been correctly understood. If the system is confident, it confirms the item implicitly by including the new information given by the user into the system response. Otherwise the system will confirm explicitly. An example of implicit confirmation is the following:

User: “I want a train to Paris.”
System: “So you want to go to Paris, where are you leaving from?”

The DM gets a semantic frame from the NLU component. This frame is filled by interpreting the utterance in the context of the ongoing dialogue, common sense knowledge, task domain knowledge, and dialogue history. The DM then prompts for missing information or sends a database query. Before the query is sent off, interpretative and history management rules are applied to determine whether new information is contained in the query and whether this information is contradictory to information given before. If this is the case then the DM can either keep the original information, replace it with the new one or engage in a confirmation sub-dialogue. The DM aims to give a direct response to the user, highlighting new information.

The main objectives of the dialog strategy are:

To never let the user get lost.

The user must always be informed what the system has understood. This is especially important for users who are unfamiliar with talking to a machine.

To use direct responses to user questions.

The systems answers should be accurate and answer to the user’s request directly.

To give the user the opportunity, at each step, to correct the system.

The user has to be able to correct errors and also to change his or her mind.

To avoid misunderstanding.

Even though users can correct the system at any time, experience in ARISE showed that the users tend not to do so. Since users cannot be relied upon to correct the system, recognition errors have to be kept to a minimum – and in ARISE it was decided to reject unreliable hypothesised words.

When errors occur, then these exceptions are handled in an explicit way, which means that the DM provides the user with clear hints on answering possibilities.

3.2 Cooperative Planning Systems

3.2.1 TRAINS

TRAINS is a uni-modal spoken dialogue system, developed at the University of Rochester. Several instantiations have been developed, such as TRAINS-93 (Traum 1996) and TRAINS-95 (Allen et al. 1996). We will not distinguish here between them, unless it is absolutely necessary and then this will be indicated.

The goal of this work was to demonstrate that it is feasible to construct robust spoken natural dialogue systems. This was done to develop techniques to enable existing theories to be applied in practice, not to develop new theories.

The domain of TRAINS is simple route planning, where a user is given a map of city connections, a list of train locations, and a list of destinations where trains are needed. The task to be solved is to find the most efficient route possible. The route planning part of the system is deliberately weak to enforce interaction between the user and the system.

An overview of the organisation of the TRAINS-95 system is shown in Figure 6. The speech recognition system is the Sphinx-II system developed at CMU (Huang et al.

1993); the speech synthesiser is the commercial product TRUETALK from Entropy. The rest of the system has been developed at Rochester.

We will concentrate here on the discourse manager, which comprises components handling references, speech act interpretation and planning (also called the verbal reasoner), the problem solver, and the domain reasoner.

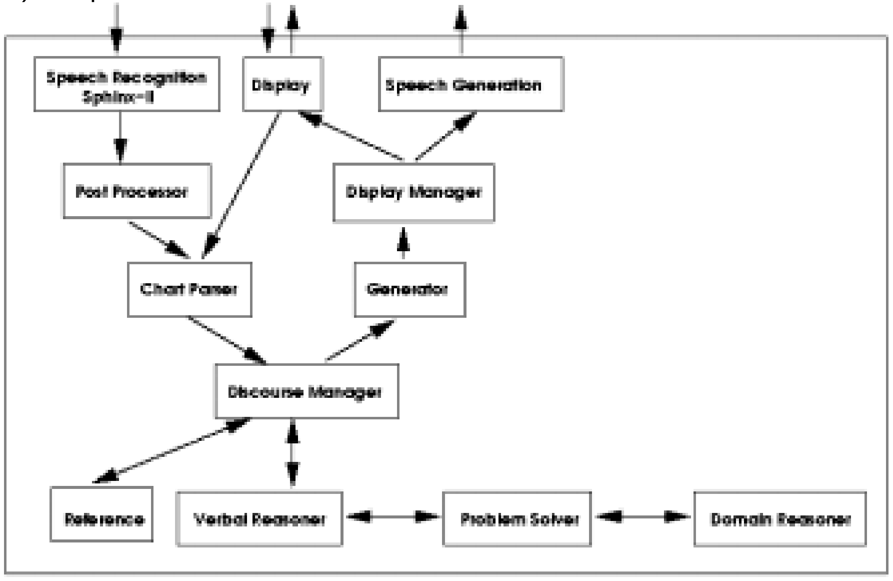

Figure 6: The TRAINS System

3.2.1.1 The Discourse Manager

The responsibility of the dialogue or discourse manager (DM) is responsible for maintaining the flow of conversation and making sure that the conversational goals are met (the main goal in TRAINS is to execute a plan which has been agreed upon by both user and system and which adheres to the constraints). In order to do this, the DM has to interpret the speech acts in context, formulate responses and to maintain a model of its mental state and the discourse context.

The model of the mental state includes:

nested beliefs and proposals about the domain;

A set of current discourse goals;

A set of intended speech acts the system will say when it gets a chance; and

A set of pending *discourse obligations* (Allen et al. 1995).

In Trains, a simple obligation model is used, based on rules which encode discourse conventions. An example rule is that an accepted offer or a promise will incur an obligation.

The DM uses he following priorities when deciding what to do next (verbatim from Allen at al. (1995)):

1. “Discourse obligations resulting from speech act effects (e.g., to address a request by accepting or rejecting it);

2. A general obligation not to interrupt the other's turn. Unless something is more pressing, the system will wait rather than interrupt when the manager has the turn;
3. Intended speech acts. When the system has the turn and doesn't have any pending discourse obligations, it will perform acts which have been planned but not yet generated.
4. A general obligation to ground expressed content (i.e., coordinate shared beliefs): Given that the system has the turn and has no obligations or speech acts it intends to perform, it should acknowledge utterances that have not been acknowledged, or request acknowledgement or repair its own utterances which have so far been ignored.
5. Discourse goals of domain plan negotiations. Given no higher priority goals, the system will address unsettled suggestions (e.g., accepting or rejections proposals previously made by the system).
6. High-level discourse goals. Given no higher priority items, the system will attempt to further the conversation, e.g. by calling the domain reasoner to expand the domain plan (in the system's private plan context), which will then provide the content for future system proposals."

The conversational state is updated when a conversation act has been perceived. This updating is asynchronous to the other operations the DM is performing. The DM will decide which task to perform next – depending on the priorities above and then work on that task. After completion, it will run through that loop again and decide on the next task. Note that the context may have changed by then, e.g., through the perception of another conversation act. The DM is always running and it decides at each iteration whether to speak or not (taking turn-taking conventions into account). It does not have to wait until an utterance is observed, and it does not need to respond to utterances in an utterance-by-utterance fashion.

One of the fundamental decision made for the TRAINS project is "that when faced with ambiguity it is better to choose a specific interpretation and run the risk of making a mistake as opposed to generating a clarification sub-dialogue." This strategy depends heavily on the system's ability to recognise and interpret corrections and a significant effort has been made to ensure that the system can detect and handle a wide range of corrections.

The reference resolution has not only the job of resolving references but also the job of reinterpreting the illocutionary act that the parser assigned, if it has additional information to draw upon.

This in important way in which robustness was achieved within the TRAINS system. Modules have overlapping realms of responsibility. One module might have more information and thus be better able to resolve a problem, still recognising another module's expertise.

3.2.1.2 The Domain Plan Reasoner

The domain plan reasoner (DPR) provides reasoning about the TRAINS domain on demand, together with a question answering facility about the current plan and the state of the world. It also has planning capabilities which is used to drive the system's own reasoning about the plan.

The DPR has to able to interpret utterances of the following types:

Suggestions of a course of action, e.g., *Send the engine E3 to Dansville*.

Identification of relevant objects to use, such as *Let's use engine E3* or *There is an Orange Juice factory at Dansville*.

Identification of relevant constraints, e.g., *We must get the oranges there by 3PM* or *Engine E2 cannot pull more that 3 carloads at a time*.

Identification of relevant lines of inferences, such as *The car will be there because it is attached to engine E1*.

Identification of goals of the plan, e.g., *We have to make orange juice*.

Introduction of complex relations, such as *Use E3 to pick up the car* or *Send engine E3 to Dansville to pick up oranges*.

To be able to interpret these different kinds of utterances, it is necessary to take a more general view of plans that the one typically found in the planning literature. In particular, the motivations for actions and explanations of why a certain action will have the desired effect have to be represented. In the TRAINS project, which was done by introducing **plan graphs**, which are graphs consisting of nodes labelled in a temporal logic (Event Based Temporal Logic or EBTL in the case of TRAINS, see Allen et al (1995) for more detail). An example of such a plan graph can be seen in Figure 7.

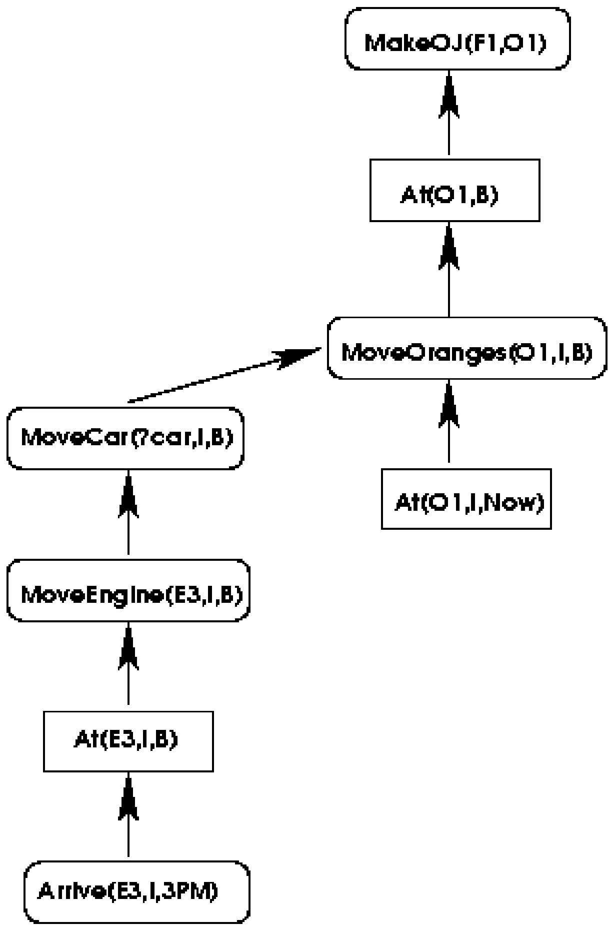

Figure 7: Example of a plan graph

There are two types of nodes, those describing events (including actions) and those describing states. The links between the nodes correspond roughly to the possible relations between events and states: **effect** (action to state), **enablement** (state to action), **generation** (event to event), and **justification** (state to state).

Plan reasoning is then defined as a search through a space of plan graphs, with the termination criterion depending on the type of reasoning to be done. Additional properties are relied upon to restrict the search of this potentially arbitrarily complex search space (for details see Allen at all (1995)).

Once a plan has been agreed upon, it will be passed to the execution planner and be executed.

3.3 Framework/Architecture Systems

3.3.1 Trindi

The Trindi project (Larsson 2000) proposes an architecture and toolkit for building dialogue managers based on an *information state* and *dialogue move engine.* The Information state of a dialogue represents the information necessary to distinguish it from other dialogues, representing the cumulative additions from previous actions in the dialogue and motivating future action.

Trindi offers a platform for the formalization of the notion of information state which allows specific theories of dialogue to be formalized, implemented, tested, compared and iteratively reformulated. Key to this approach is the notion of UPDATE of information state with most updates related to the observations and performance of DIALOGUE MOVES.

The Information State Theory of Dialogue Modelling consists of

A description of the **informational components** of the theory of dialogue modelling including common context and internal motivating factors(common ground, commitments, beliefs, intentions, etc.)

Formal representations of the above components (e.g. as lists, sets, typed feature structures, records, etc)

A set of **dialogue moves** that will trigger the update of the information state (also correlated with externally performed actions such as particular natural language utterances).

A set of **update rules** that govern the updating of the information state given various conditions of the current information state and performed dialogue moves including a set of selection rules.

An **update strategy** for deciding which rule(s) to select at a given point from the set of applicable ones.

There is an important distinction between information state approaches (Cooper and Larsson 1999) and dialogue state approaches. In a Dialogue state approach, a dialogue behaves according to some grammar where the states represent the results of a dialogue move in a previous state and each state has a set of allowable moves. The “information” is implicit in the state itself and the relationship it plays to other states.

The **informational components** are not conceived of as monolithic nodes in a transition network (as with dialogue state, but rather as consisting of several interacting components). One could model the mental state of an agent or take a more structural view and model the performance of actions. The **formal representations** for modelling various aspects of the dialogue structure range from simple abstract data types to more complex informational systems such as logics

Dialogue moves

There are a number of different dialogue move taxonomies to choose from.
Dialogue moves do not need to be conceived of as *speech-acts* (Searle 1969), but can be any mediating input, e.g., logical forms or even word-lattices augmented with likelihoods.

Update rules Example

```
U_RULE: integrateSysAsk
PRE: val(SHARED.LM, ask(usr, Q))
       Fst(PRIVATE.AGENDA, raise-question(Q))
EFF:             push(SHARED, QUD, Q)
                 Pop(PRIVATE.AGENDA)
```

Rule for adding a question to QUD (Questions Under Discussion) if an *ask* move is performed. This rule has 2 conditions: 1) that the latest move was of type *ask* and 2) that the top of the agenda was the action of raising a question. The effects are to pop this item from the agenda and push the question that is the content of both the raise agenda item and the *ask* dialogue move.

Update Strategy

Some types of update strategies:

Take the first rule that applies (iteratively until no rules apply)

Apply each rule (if applicable) in sequence

Apply rules according to class

Choose among applicable rules using probabilistic information

Present choices to user to decide (for development modes)

TrindiKit: A Dialogue Move Engine Toolkit

The implementation of the above theory of dialogue dynamics is called a **Dialogue Move Engine** (DME) since its main functions are updating the information state based on the observance of moves and selecting moves to be performed. This DME, together with some connective material, forms the dialogue management and discourse tracking aspects of a dialogue system. A complete dialogue system needs modules for additional functionality
:

User interface – I/O

Interpretation – to calculate which dialogue moves are performed

Generation – to take the next move part of the information state and produce the output.

Control – to link together the other modules either serially or in parallel.

The Dialogue Management Architecture has 3 parts:

Control module

The Dialogue Move Engine

Contains one or more Update modules each containing a different set of rules and potentially different algorithms.

Information State

Apart from the basic structure of these 3 modules, a system may also include resources such as databases, plan libraries which are accessible using an interface that allows the same kinds of queries and operations as for the IS (Information State) proper.

Implementations using TrindiKit

GoDiS – An experimental dialogue system built using fairly simple algorithms for control, update and selection modules, keyword-based interpretation and template-based generation. This system distinguishes 8 dialogue move types: ask, answer, repeat, request_repeat, greet, goodbye, thank and quit.

EDIS- This system (Matheson et al 2000) uses a notion of information state based on (Poesio & Traum 1997) using the record representation for coding information states in (Cooper et al. 1999; Poesio et al. 1999)

MIDAS system uses the DRS structures of DRT (Kamp & Reyle 1993) as a major component of the information state.

A system that focuses on conversational games theory, recasting the existing **Autoroute** system (Lewin 1998) in the current framework

Contributions of TrindiKit

1. This is a toolkit for building dialogue managers that can be closer to the level of a theory of dialogue processing than is available in most systems. Bridges the gap between practical dialogue systems and more theoretical approaches to dialogue.

2. They hope that the specification of components of information state can help clarify both what is meant by dialogue management and the role of different aspects of dialogue management. For example, in many systems, the key dialogue management task is control of the dialogue. In other systems (Allen et al. 1996), there is a central hub controller and the dialogue manager is one of many "spokes". With clear separations between information state, update mechanisms and control flow, the framework that Trindi offers can help clarify some “tricky issues”. Lastly, one of the main advantages of implementing dialogue systems in this framework is that it provides the ability to compare different systems or approaches to the same phenomena.

3.3.2 Galaxy Communicator

The Galaxy communicator software infrastructure (URL) is a distributed, message-based, hub-and-spoke infrastructure which is optimised for constructing spoken dialogue systems. It is an extension and evolution of the distributed infrastructure for the MIT Galaxy system (url) and it is being developed and maintained by the MITRE corporation under the DARPA Communicator Programme. An example system, which can be built with the Galaxy Communicator Software Infrastructure (GCSI) can be seen in Figure 8.

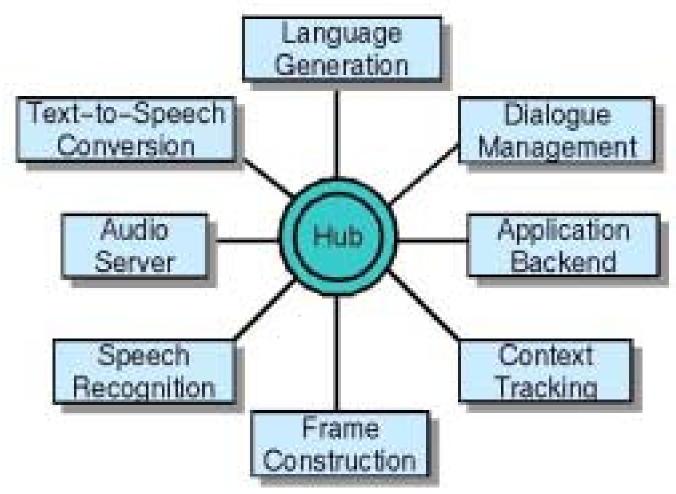

Figure 8: General Architecture using Galaxy

A number of systems have been developed using the GCSI, some of which will be briefly described in the final version. We will concentrate on the dialogue management part of the systems.

The systems are:

the CMU communicator system (Rudnicky et el., 1999; Xu and Rudnicky, 2000);

the CU communicator and CU move, University of Colorado (Pellom et al., 2001);

Mercury, MIT Spoken Language Systems (Seneff and Polifroni, 2000);

SRI communicator (Walker et al., 2000).

3.3.3 DISC

The DISC project on "Spoken Language Dialogue Systems and Components: Best practice in development and evaluation" was initiated by ELSNET and launched on 1 June 1997. It lasted until 31 December 1998 but was immediately continued in the DISC-2 project which stopped on 31 December 1999.

The first phase of DISC consisted in developing a detailed set of development and evaluation methods and procedures for best practice in the field of dialogue engineering as well as a range of much needed dialogue engineering support concepts and software tools.

The second phase of DISC, i.e. DISC-2, was mainly focused on testing the validity and the usability of the draft best practice scheme, the concepts and the tools, and on the integration, packaging and dissemination of the final best practice scheme.

The DISC Best Practice Guide extends and specialises software engineering best practice to the particular purposes of dialogue engineering, that is, to the development and evaluation of spoken language dialogue systems (SLDSs).

The Guide consists of the DISC Dialogue Engineering Model, a series of SLDS design support tools, state-of-the-art overviews and references to relevant sites and running SLDSs world-wide, a glossary, DISC publications, and Guide evaluation stuff for you to use.

3.3.3.1 DISC Dialogue Engineering Model

The following six aspects of SLDS are addressed by the DISC dialogue engineering model:

speech recognition,

speech generation,

natural language understanding and generation,

dialogue management,

human factors, and

systems integration.

Each of these broken down into **grids** and **life-cycles**. Grids address factual properties of SLDSs and components, inter-relationships between the properties, and advice on which properties to use for which applications. The life-cycle addresses the development process of an SLDS or component as well as the evaluation.

In the following section we describe the dialogue management grid, one of the six grids for the six aspects mentioned above.

3.3.3.2 Dialogue Management Grid

A set of issues should be considered when developing the dialogue management component of an SLDS:

Goal of the Dialogue Manager (DM),

System varieties,

Input speech and language,

Getting the user's meaning,

Communication, and

History, users, implementation.

The central **goal** of the DM efficient task performance, with the sub-goals of robustness, flexibility, naturalness etc. The **system varieties** to think about include multi-modal systems (including speech), multi-lingual systems, and the question whether there is one or several tasks and users. Questions regarding the **input speech and language** include questions like Are the speech and language layers ok? Do the speech and language layers need support from the DM? And questions about real-time requirements. **Getting the user's meaning** includes the issues of the task complexity, controlling user input, who has the initiative, input prediction, sub-task identification, and advanced linguistic processing. **Communication** involves domain communication, meta-communication, other forms of communication, expression of meaning, error loops and graceful degradation, feedback, and closing the dialogue.

3.4 Tutorial Systems

In this section we present several dialogue management systems that are part of tutorial systems.

3.4.1 Tutorial Dialogue in Shipboard Damage Control

This system is based on DC-TRAIN for training damage control assistants to manage shipboard crises appropriately (Bratt 2002). The DC-TRAIN system involves a simulation of a destroyer, its firemain and the properties of its compartments with respect to fires.

The system architecture contains:

- Gemini – natural language understanding (unification grammar)
- Nuance speech recognition system
- Festival speech synthesis module
- Dialogue Manager
- Tutoring Module which identifies appropriate tutorial strategies

The Dialogue Management Architecture has

An Information State corresponding to dialogue context, which includes 2 types of declarative knowledge

- Dialogue move types
    - o Java classes including assertion, wh-query, yes-no-response, expand_only (does not require a response), summary
- Activity model
    - o Hierarchical and temporal decomposition of tutorial activity

The Dialogue Manager includes the following dynamically updated components:

- A *Dialogue Move Tree*: a structured history of dialogue moves and threads, plus a list of 'active nodes'.
- An *Activity Tree*: a temporal and hierarchical structure of activities initiated by the system or the user, plus their execution status.
- A *System Agenda*: the issues to be raised by the system
- *Salience Groups*: the objects referenced in the dialogue thus far, ordered by recency (Fry et al. 1998)
- *Pending List*: the questions asked but not yet answered
- *Modality Buffer*: stores gestures for later resolution

In the Dialogue Move Tree, each node is of a particular dialogue move type either by the tutor or the student.

In the Activity Tree, each node is of a particular activity type. Nodes are constructed and added to the tree by the Tutoring Module. At the initial part of the session, the system creates a single EXPAND_ONLY node called start and adds it to the tree. It then begins executing tasks. The activity 'start' is passed to a hashtable which returns the corresponding sequence of actions. Activities are added to the tree in accordance with the dialogue.

Benefits

- The dialogue management architecture is reusable across domains.
- The Dialogue Tree/Activity Tree distinction allows one to capture the notion that dialogue works in service of the activity the participants are engaged in. That is the structure of the dialogue is a by-product of the Activity Model and the Tutoring Module.
- The dialogue move types are domain-general and reusable in other domains.

The architecture supports multi-modality with the Modality Buffer.

3.4.2 Beetle

The system called Beetle is a prototype implementation of a computational framework (Zinn 2002). It uses TrindiKit and OAA.

Modular dialogue system architecture

Interpretation module – interaction via text and graphical means. This module identifies the meaning and intention behind an utterance. This includes syntactic analysis, the construction of its propositional content, speech act recognition, intention recognition (dialogue act)

update module – This module maintains the context. Update rules encode conversational expertise and define how to update the current context given a new dialogue act. This module takes in the dialogue move and fires the corresponding rule. The move is added to the current discourse unit (CDU) and an obligation for the hearer to address this move is created.

response generation module- This module uses a 3-tier planning architecture to compute the tutorial moves and synthesises tutorial feedback as text or other modalities. The three tiers are as follows:

- *Top-tier : Deliberative Planning*- This module works at the highest possible level of abstraction. High-level planning results in a structured sequence of tasks that are passed to the *task agenda* of the middle layer. For each task, there is a reactive action package that achieves the goal. The Deliberator is activated in 2 cases: when the tutor agent enters a tutorial dialogue and during the dialogue when the middle layer fails to perform plan execution and needs to ask the top layer for a plan repair. The top layer is always inspecting the agenda of the middle-tier to 1) verify that pending discourse obligations (recorded in the IS (Information State) are covered by the contents of the agenda and 2) searches to anticipate problems/optimise the agenda's content.
- *Middle-tier : Context-Driven Plan Refinement* – This is situation–driven plan refinement that is carried out through Reactive Action Packages. RAPS group together and describe all ways to carry out a specific task in different situations. The RAP interpreter executes the contents of an agenda by selecting the next item on the agenda and checking the IS to ensure that the actions haven't already been carried out. If the actions have not been completed, the interpreter identifies the RAP that can achieve the task. The most appropriate *method* of the RAP is chosen. If the method is a primitive action, it is delegated to the bottom tier/execution layer. If the method is a group of subtasks, each subtask is put on the agenda and a new interpretation cycle begins. Components of RAP are *preconditions, effects, and methods*. *Methods* include a *context* and *list of tasks.*
- *Bottom-tier : Action Execution* – Responsible for the execution of primitive dialogue actions. It gets as input a sequence of elementary speech acts and plans the generations of multi-modal feedback (speech + GUI actions). This module is supported by a sentence and media planner which have access to the full dialogue context.

Turn-taking – The tutoring agent releases the turn after it asks a question or requests an action. Otherwise the RAP interpreter continues to 'cycle'. If the student seizes the turn, this action is recorded in the IS and the top-tier deliberates over the new situation and changes the agenda accordingly.

Domain reasoning is performed by a single agent. Each of the modules can access the information state which captures the overall dialogue.

3.4.3 AutoTutor

The goal of this project is to build an agent that delivers pedagogically effective dialogue moves which participating in a conversation with the learner. AutoTutor (Person and Graesser 2000) is a fully automated computer tutor that helps college students learn about introductory computer literacy topics. In a typical tutoring session, the student learns the fundamentals of computer hardware, the operating system and the Internet. The interface to AutoTutor is made up of four features

An animated embodied agent,
A text box for the student input
A text box that displays the problem/question
A graphics box that displays pictures/animations related to the text topic

How it works:

The architecture is comprised of 5 major modules

An animated agent
A curriculum script
Language analysers
Latent semantic analysis (LSA)
A Dialogue move generator

Dialogue Move Generator is governed by 15 fuzzy production rules (Kosko 1992) that use data provided by the LSA module. Each fuzzy production rule specifies the parameter values in which a particular dialog move should be generated.

Example: If [Assertion match with good answer bag= HIGH or VERY HIGH] THEN [select POSITIVE FEEDBACK]

There are 11 dialogue moves – pump, hint, splice, prompt, elaborate, summarize and 5 forms of immediate short-feedback (positive, positive-neutral, neutral, negative-neutral, and negative).

Dialogue Advancer Network seems to be implemented as a Finite State Machine. The DAN assigns student contributions to different speech act categories (Assertion, Question, Short Response) and then generates discourse markers and dialogue moves accordingly. Each dialogue move production rule specifies conditions for which a certain dialogue move should be generated. Auto-Tutor is able to sustain mixed-initiative dialogue.

AutoTutor's Conversational Features

Indicate when the student has the floor to contribute (turn-taking)
Adapt each dialogue move to the previous turn of the student. (coherence)
Deliver dialog moves that are consistent with conversational norms (Follows Gricean maxims of conversational norms)
Appear interested in what the learner is saying (Uses backchannel feedback mechanism)

3.5 Other Systems

3.5.1 VerbMobil

The dialogue processing component of VERBMOBIL (Alexandersson and Reithinger 1995), a speech-to-speech translation system, needs to cope with unexpected and vague input as well as gaps. The resulting architecture consists of a hybrid approach using both knowledge-based and statistical processing. The users of the system interact in English and activate VERBMOBIL only if the owner of VERBMOBIL lacks sufficient knowledge of English and demands the translation of utterances in her mother tongue.

The application is an appointment scheduling dialogue between two business people, one of them German, where both are non-native speakers of English. If the German partner is not able to express himself adequately he can switch to his mother tongue indicating the need for translation by pressing a button. VERBMOBIL is then expected to translate the utterances into English.

The Corpus
The speech data corpus gave rise to 17 different dialogue acts, defined together with semi-formal rules for their assignment to utterances. 200 dialogues were annotated with dialogue act information and a hierarchy

1. A hierarchy of dialogue acts was developed where the upper concepts are domain independent and the lower concepts are more domain dependent.
   Example: dialogue act reject reject_date, reject_location, reject_duration
2. From the analysis of the annotated corpus, a model of dialogue act sequences was derived.

The tasks of a dialogue component

To provide and to store contextual information which is used by the linguistic modules of VEBMOBIL e.g. the transfer component.
To provide top-down expectations about what dialogue steps are most likely to follow. This information is used to support the analysis components for narrowing down the search space (important for speech processing systems)
To integrate both modes of processing within a unified approach to get a hold on the overall flow of the dialogue

The Internal Structure

To store the contextual information, they follow the approach of (Grosz and Sidner 1986) for modelling the context and describe it by three interconnected knowledge sources
An intentional structure containing information about the intentions for parts of the dialogue.
A thematic structure which represents local and global focus and the development of different topics in the dialogue.
A referential structure which links the conceptual and language-related information for the objects mentioned.

Implemented as an automaton

Verbmobil has no control over the dialogue steps.
They divided processing up into two parts

1. One part is responsible for building up the intentional and thematic structure.
2. One for the prediction process.
   Architecture - contains three components
   1. Plan recognizer,

2. Finite state machine
3. Statistics

Finite State Machine

Uses the dialogue model as aknowledge base

Parses the incoming dialogue act and checks whether the ongoing dialogue is in accordance with the dialogue structure as defined in the networks.

If the dialogue act is not within the language of the model, the FSM uses a fall-back strategy based on the information in the statistics component to find the most probable follow-up state.

They tried to use the FSM to predict the next dialogue act to come, but the results were poor.

The Statistics Component

To provide weighted predictions for the most probable follow-up speech acts, and used a statistical approach.

The method is based on n-gram speech act probabilities, a method adapted from speech recognition.

To approximate the conditional probability, they used a technique known as deleted interpolation.

They have shown that a training corpus of about 50 dialogues is sufficient to get an approximation based mainly on bigrams.

Plan Recognizer

Taken from (Vilain, 1990), the first version is a simple top-down parser with backtracking where the plan operators are processed strictly left-to-right (mimicking a Prolog interpreter).

The plan recognizer operates on a set of plan operators. In this system, they are the rules forming a grammar. The rules encode both the dialogue model and methods for recovery from erroneous dialogue states. Each rule represents a specific goal which can be fulfilled if the constraints for the rule hold. Plan operators can trigger follow-up actions and can define subgoals which must be achieved in a pre-specified order to fulfil a goal. Because the parser either accepts or rejects the input, a method was developed to handle the reject situation so the module does not fail completely. The two methods are:

A method that relies on the information in the statistical component allowing for reinterpreting of the input. This method is based on the hypothesis that multiple speech acts can be attributed to an utterance. They try to bridge the previous utterance with the current utterance by finding a relevant dialogue act

This method uses a set of *repair operators* for *repairing* the parse tree. They include a separate subnet in their model which is activated in this case (Clarify_Query, Clarify_Answer, Give_Reason, Deliberate, Digress) rather than including this as part of each dialogue state.

Summary of the design process for the dialogue component

Annotate a corpus

Extract a "standard" dialogue model from the annotations

Check the requirements from the other components in the system and identify information needed from the dialogue component

Select appropriate processing methods, in their case plan-based and statistical approaches which are combined for robustness.

Evaluate the system

Tune the system again using real data.

3.5.2 WITAS

This system (Lemon 2001) contains a dialogue interface for multi-modal conversations to the WITAS robot helicopter. The requirements of this dialogue system are:

Asynchronous
Mixed-Initiative
Open-ended
Involves a dynamic environment

Dialogue System Architecture

Speech Recogniser (Nuance)
Natural Language Processor (wrapper to Gemini Parser and generator)
Text-to-Speech (wrapper to Festival speech synthesiser)
GUI (a map display of the current operating environment which displays route plans, waypoints, locations of vehicles etc.).
Dialogue Manager

- coordinates multi-modal inputs from the user
- interprets dialogue moves made by the operator and UAV (Unmanned Aerial Vehicle)
- updates and maintains the dialogue context
- handles UAV reports and questions
- sends speech and graphical outputs to the operator

Robot Control and Report – translates commands and queries from the dialogue interface into commands and queries to the UAV and vice-versa for reports and queries from the UAV.

The Dialogue Manager (version I)

Creates and updates an Information State corresponding to a notion of dialogue context.

Dialogue moves have the effect of updating information states and moves can be initiated by both the operator and the robot.

A dialogue move might:

Cause and update to the GUI
Send an immediate command to the UAV
Elicit a spoken report
Prompt a clarifying question from the UAV

The Information States consist of:

Issues Raised stack
UAV AGENDA
Salience list
Modality buffer
Databases (dynamic objects, planned routes, geographical information, names)

They found that a stack structure was too restrictive because navigation back and forth between different sub-dialogues and topics was impossible because information was lost when stack was popped. To compensate for this, they implemented Version II of the dialogue management system which uses a tree structure of dialogue states (*dialogue move tree*), where edges are dialogue moves and branches represent conversational threads.

They also wanted to enrich their domain knowledge and inference methods so they implemented a dynamic hierarchical *task tree.* The *task-tree* grows as part of the developing dialogue context and represents tasks and sub-tasks described by the operator and their temporal ordering. This structure allows for reordering and reference to tasks.

They also implemented an inference-based model of the robot’s changing abilities, depending on dynamic information about the world and the robot’s internal state and location. (carried out using KIF knowledge representation scheme and the Java Theorem Prover).

Note: Uses the same dialogue manager as the CSLI Tutoring system.

3.5.3 CONVERSE

CONVERSE (Levy et al. 1997) was a machine dialogue system funded by Intelligent Research of London which won the Loebner prize in 1997 (and did not enter again against those it had beaten before in that year). That year was the first in which there was no restriction on the topic of discussion with judges, and CONVERSE covered about 80 topics, which were appropriate to its persona as a young female New York-based journalist. It embodied substantial resources, such as WordNet, the proper names of Collins dictionary etc. It could store the Personal information it elicited from a user and build it into the conversation later. Its topic structures were complex ATN scripts that could be left and reentered appropriately and could generate responses using stored/elicited material.

It had no conventional analysis/DAM/ generation division, though it used a commercial statistically based parser to pass input to the ATN's. Its control structure was simple blackboard system in which the ATN's competed to take control of the generation; these decisions were made numerically based on weights assigned by the closeness of fit of the input to the expected input etc. The system had only limited recovery mechanisms if it was not able to find a topic relevant to the input, and relied of seizing control of the conversational initiative as much as it could. Since this system models only plausible conversation, the dialogue had no application goals of any kind.

4 Conclusion/Recommendation

The preceding survey suggests that we may now be, in dialogue systems, be in something of the same position as the field of Information Extraction (IE) when Jerry Hobbs wrote his brief note on a generic IE system, based on the assumption that all functioning IE systems contained roughly the same modules doing the same tasks, and that the claimed differences were largely matters of advertising and dubious claims to special theories with trademarked names. However, we are in a worse position with dialogue systems because, unlike IE, there is little or no benchmarked performance with which to decide which modules and theoretical features aid robust dialogue performance.

Again, it is hard to derive much benefit from taxonomies of dialogue systems--though we have of necessity deployed one above---because, as we noted there, even systems which might seem in the most primitive category also lay claim to features from the higher ones: Carbonell's POLITICS (Carbonell 1981) seems just a series of questions and answers to a complex knowledge base and might therefore be deemed a very simple dialogue grammar, lacking even a global strcuture of greetings and goodbyes. But it is clear that he considers it to deploy coded forms of goals, beliefs and plans, which we are taking as a necessary property for a more developed class of systems.

Again, PARRY (1973) the most developed and robust of the early dialogue systems, might seem to be simply a dialogue grammar at heart, yet it very clearly had goals of informing the user of certain things, and even though it had no explicit representation of goals and beliefs, it did have a primitive but explicit model of the user.

Even so, we can move ahead by assuming that a plausible DAM must draw functionality from all three of those categories of systems and must at least do the following:

a) determine the form of response locally, where appropriate, to dialogue turn pairs, where appropriately means in both pragmatic (i.e. dialogue act functional) and semantic terms (i.e. give correct answers if known to questions).

b) have some form of representation of a whole dialogue, which means not only opening and closing it appropriately, but knowing when a topic has been exhausted, and also how to return to it, if necessary, even though it is exhausted from the system's point of view. This functionality need not imply an explicit global representation or 'grammar' of a dialogue.

c) have appropriate access to a data base if there is to be question answering on the basis of stored (usually application relevant) knowledge.

d) have appropriate access to a database that can be populated if information is to be elicited from the user as a basic task.

e) have some form of reasoning, goal/intention representation, user modelling and planning sufficient to perform these tasks, though this need not imply any particular explicit form of representation or mechanism for these functionalities.

f) have some general and accessible notion of where in the performance it is at any moment: this can take the form of where in the dialogue or where in the overall task performance (if there is a task) or both. The only exception to this would be Loebner type systems whose only function is to keep talking coherently and not repeat themselves.

Not all the above are essential, since some depend on the overall application/functional environment: e.g. (c) above is needed only for systems that know something and can answer questions about it, and (d.) above applies only to tasks that can be captured as 'form filling'. Our application, and most conceivable ones, requires both those. A trickier matter will be (f.) where it is by no means agreed what applications require explicit models of users and of their goals, beliefs and intentions etc. Again, the power of any reasoning system implied by (f.) is by no means clear, given both the certainty, from Church, that inconsistencies cannot in general be detected as well as the overall mediocre performance of general AI reasoning systems.

Our proposal in this most complex and inviting area will be to so design the DAM that minimal systems can serve some of these functions initially but without preventing the substitution of richer ones later on. At every stage, our prejudice will be in favour of limited, constrained, knowledge-based systems, rather than general inference mechanisms of unconstrained power.

The functionalities above will probably not give rise to much dispute, nor will the sorts of module in a DAM to implement them, at least not if the Hobbs-vanilla assumption at the beginning is broadly right. However, something must be said about other desiderata and constraints (before processing to a DAM sketch), ones that follow both from our own theoretical prejudices and from the global design of COMIC within which we now have to work.

The global design implies (unsurprisingly) what one might term an (almost) fully interlingual DAM: one in which the DAM deals only with coded representations of content and function as input and output (i.e. "first order English" more or less) and not with language strings (in German in this case). We write "almost" because we will get the German input string to the DAM (though we do not of course want to attempt a second parsing of it) but not the German output string, which is the job of Fission to create (or its other-modal equivalent). However, this conventional (indeed archaic) modularization of the DAM leads to two well-known problems, which will give rise to lively discussion between the DAM and Fission groups, we imagine.

g) (*We will continue to label key issues alphabetically, even though they are not ALL of the same type*) There is the issue of not knowing the output string, and the problem that E.M.Forster expressed as "How do I know what I think till I see what I say". DAM does not know what COMIC says so how can it know what it thinks?

In a sense this is absurd, at least in terms of AI/NLP orthodoxy, but there are the serious attendant problems what we will later call the "information retrieval problem"---how do we find the relevant task/response structure without the surface information in input and potential output? ---as well as the psychological problem for researchers, known since the 1960s and immortalised in McDermott's paper (McDermott 1981), that NLP researchers cannot work with, or even create, structures they cannot understand and structures in first order English may or may not be comprehensible to the DAM team, particularly if made up by another (Fusion team) working in another natural language. Let's call this just a minor development problem.

The DAM is thus in some sense a Chinese Room, receiving and emitting coded forms it may not understand, and that may cover the researchers as well as the algorithm. The solution is obvious and we shall just announce it as a principle:

h) The DAM team will decorate structures and rules with strings of arbitrary length consisting of English and German words, for both retrieval and mnemonic functions. These may in many cases correspond to the most plausible generated output in English or German or both and these will be passed on to Fission who can do with them what they like. It should be obvious that retrieval of relevant action frames is more likely with a wider set of retrieval terms/wider than the internal predicate set). An important COMIC global issue for further discussion (that is only hinted at here) is how the Fusion, Dam and Fission teams are to understand recursive expressions in the first-order content language in the same way, as developers and as consumers of those codings.

Another assumption follows naturally here, and if accepted will have considerable consequences for Fission:

i) the format and range of slots to be filled in the interface objects that pass from DAM to Fission should be identical to those that pass from Fusion to DAM. In some sense this is obvious, and anyone who dislikes it should produce convincing cases where it is inadequate, which will mean finding information types beyond the standard Fusion-to-DAM set of (roughly): semantic content, dialogue act, (possibly) intention form, language string, modality. It may well be inappropriate to pass the latter from DAM-to-Fission except as a record of the original input (and unless that is independently unavailable to Fission from the dialogue history itself).

None of this prolegomena reaches the core of the DAM design, but just (re)states assumptions, many of them obvious. We can now add some further ones, not normally discussed in the literature much but well-known to those who have had to produce working systems, particularly those subject to evaluation or assessment, as we have.

The key problems for dialogue system performance, and therefore reasons for failure, are:

j) inability to find the relevant structure/frame that encapsulates what is known to the system about the subject under discussion, normally one introduced by user initiative. This is the main form of what we referred to earlier as the information retrieval problem in dialogue management and, unsurprisingly, we shall use IR methods to solve it, by using, as one main determinant of what structure to load as the current structure, overlap of terms between input and (our own indexing of potential) output. The same method will be asked to provide constant confirmation that the current structure is the appropriate one, over and above considerations from heuristics expected input and global/local dialogue act sequencing.

This last can be thought of as not-getting-lost or knowing-where-we-are, yet it is not a problem that can be solved by any single consideration or simply assigned to any magic module. Another problem that may be no more than (j) under another description is that of knowing-what-to-do-now:

k) this is a problem central to all systems, and quite different strategies are used by different systems: e.g. the Rochester-style strategy (Allen and Perrault 1980) of the system taking a definite, and possibly wrong, line with the user, resting on robust measures for revision and recovery if wrong, as opposed to a hesitant (and potentially irritating) system that seeks constant confirmation from the user before deciding on any action. We shall opt for the former strategy, and hope for sufficiently robust recovery, while building in implicit confirmations wherever appropriate.

But the issue here is narrower than simply one of having additional modules: it requires a core dialogue engine that is both a simple and perspicuous virtual machine (and not a lot of data/links and functionalities under no clear control) and which can capture (given good data structures) the right compromise between push (user initiative) and pull (system initiative) that any robust system must have. Our initial sketch below is intended above all to capture this combination of perspicuity (for understanding the system and allowing data structures to be written for it) and compromise.

An initial modest proposal.

l) We propose a single pop-push stack architecture that loads structures of differing complexities but whose overall elements are ATNs at most (though this power will not always be needed). The algorithm to work such a stack is trivial and well understood, though below we will need to suggest one amendment to the classical algorithm to deal with a revision problem that cannot be dealt with by structure nesting.

m) The argument for such a structure is its combination of power, simplicity and perspicuity (we used it in the simple mini-CONVERSE system designed for an interactive child's toy). Its key language-relevant features (known back to the time of Woods (Woods 1979) in parsing practice) are:

n) the fact that structures can be pushed down to any level and re-entered via suspended execution allows nesting of topics and activities like barge-in and revision with a smooth and clear return to unfinished materials and topics. This is so well known it has entered the everyday language of computer folk as "stack that topic for a moment".

o) the ATN structures (to be stacked) have Turing Grammar/Turing Machine power and are subject to no limitations of a finite state sort: i.e. a command on a transition can do anything from filling a data slot to causing a new structure to be pushed; they can easily be seen as “update rules”. They are also perspicuous to write and understand (given our ability in our Chinese room to make any notes on our structures we like).

Formal power is a diversion here: it may well be the case that, for efficiency, some later COMIC should have all its structures reimplemented as finite state rules or graphs. But that is irrelevant here, since such structures, though easy to write, are hard to understand in the absence of any context dependence.

p) There will be such ATNs corresponding to each of the system-driven sub-tasks (i.e. form filling--the form the bathroom salesman aims to end up with at the end of a session) which are for information eliciting (and whose ATN commands write to the output database), as well as for standard Greetings and Farewells, and as well as for complex tasks like revisions and responses to conversational breakdowns. Mini-ATNs will express simple dialogue act pairs (such as QA) which can be pushed at any time (from user initiative) and will be exhausted (and Popped) after an SQL query to the bathroom database.

We propose that the stack be preloaded with a (default) ordered set of system initiative nets, with Greeting at the top, Farewell at the bottom and such that the dialogue ends with maximum success when all these are popped and all the information eliciting networks have been popped. This would be the simplest case of a maximally cooperative user with no initiative whatever; he may be rare but must be catered for if he exists!

An obvious problem arises here, which may require that we adapt the overall DAM control structure:

q) If the user proposes an information eliciting task before the system does (e.g. wants to discuss tile-colour-choice before that structure is reached in the stack) then that structure must be pushed into the stack and executed till popped but obviously its homologue in the stack must not be executed when it reaches the top. The integrity of the stack algorithm need be violated here only to the extent that any task-driven structure at the top of the stack is only executed if the relevant part of the database is empty.

However, a closely related issue (and one that caused the WITAS researchers to change their DAM structure, wrongly in our view) is the situation where a user-initiative forces the revision/reopening of a major topic already popped from the stack; i.e. the user chooses pink tile but later, and at her own initiative, decides she would prefer blue and brings up the topic again. This causes no problems: the tile-colour-choice structure is pushed again (empty and uninstantiated) but with an entry subnetwork (no problem for ATNs) that can check the data-base, see it is non-empty, and begin the subdialogue in a way that shows it knows a revision is being requested. It seems clear to us that a simple stack architecture is proof against simple arguments based on the need to revisit popped structures.

The above is no more than a basic sketch for discussion. What is not touched upon is the provision, outside the main stack and content-structures, of DAM modules that express models of the users goals/beliefs/intentions and which reason over these. We shall postpone this as inessential for getting started provided what we ultimately propose can transition from simpler to more complex structures and functions. At this stage, we do not believe anything as complex as the ViewGen system (Wilks and Ballim 1989) will ever be required for an application of this type, though it might well for the Nijmegen game application, SLOT. To use it for bathroom advice would require an implausible scenario where the advisor has to deal with more than one client, possibly talked to separately so that the system has to construct the couple's views of each other's wishes. We do not believe we should start by taking account of the needs of this scenario, fun though it would be for research.

5 Acknowledgement

We would like to thank the COMIC consortium for their input to and feedback of this document.

6 References

Alexandersson and Becker 2001	J. Alexandersson and T. Becker, *“Overlay as the Basic Operation for Discourse Processing in a Multimodal Dialogue System”*, in Proceedings of the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Seattle, August 2001.
Alexandersson and Reithinger 1995	J. Alexandersson and N. Reithinger, *“Designing the dialogue component in a speech translation system”*, in Andernach et al. 1995.
Allen and Perault 1980	J.F. Allen and C.R. Perault, "*Analyzing Intentions in Dialogues*", in Artificial Intelligence, 15(3):143-178, 1980
Allen at el. 1996	J.F. Allen, B.W. Miller, E.K. Ringger, and T. Sikorski, *“A Robust System for Natural Spoken Dialogue”*, in Proceedings of the 1996 Annual Meeting of the Association for Computational Linguistics (ACL'96), June 1996. pp. 62-70.
Allen et al. 1995	J.F. Allen, L.K. Schubert, G. Ferguson, P. Heeman, C. Hee Hwang, T. Kato, M. Light, N.G. Martin, B.W. Miller, M. Poesio, and D.R. Traum, *“The TRAINS Project: A case study in building a conversational planning agent”*, in Journal of Experimental and Theoretical AI (JETAI), Vol. 7, pp. 7-48, 1995.
Andernach et al. 1995	J.A. Andernach, S.P. van de Burgt, and G.F. van der Hoeven (eds), "*Corpus-Based Approaches to Dialogue Modelling*", in Proceedings of the 9th Twente Workshop on Language Technology, University of Twente, Enschede, Netherlands, 1995.
Appelt 1985	D.E. Appelt, "*Planning English Sentences*", Cambridge, University Press, 1985.
Baggia et al. 1999	P. Baggia, A. Kellner, G. Perennou, C. Popovici, J. Sturm, and F. Wessel, *“Language Modelling and Spoken Dialogue Systems - the ARISE experience”*, in Proceedings of Eurospeech '99, Budapest, Hungary, 1999.
Beun 1996	R.J. Beun, *"Speech Act Generation in Cooperative Dialogue"*, in Luperfoy et al. (1996).
Bontcheva, 2001	K. Bontcheva, *"Generation of Adaptive Hypertext"*, PhD thesis, Department of Computer Science, University of Sheffield, 2001.
Bratman's et al. (1988)	M.E. Bratman, D.J. Israel, and M.E. Pollack, *"Plans and Resource-Bounded Practical Reasoning"*, in Computational Intelligence, 4, 1988

Bratt 2002	E. O. Bratt, B. Clark, Z. Thomsen-Gray, S. Peters, P. Treerapituk, H. Pon-Barry, K. Schultz, D. Wilkins, and D. Fried, *“Model-Based Reasoning for Tutorial Dialogue in Shipboard Damage Control“*, at ITS 2002, San Sebastian, Spain. 63-69, 2002 .
Bunt 1989	H.C. Bunt, “*Towards a dynamic interpretation theory of utterances in dialogue”*, in Elsendoorn and Bouma 1989
Carbonell 1981	J. Carbonell, "Overview of Politics", in Schank (1981).
Carletta et al. 1995	J.H. Carletta, A. Isard, J. Kowtko, G. Doherty-Sneddon, and A. Anderson, "The Coding of Dialogue Structure in a Corpus", in Andernach et al. (1995).
Churcher et al. 1997	G.E. Churcher, E.S. Atwell, and C. Souter, "Dialogue Management Systems: a Survery and Overview", Report 97.06, University of Leeds, February 1997.
Chu-Carroll 1996	J. Chu-Carroll, *"Response Generation in Collaborative Dialogue Interactions"*, in Luperfoy et al. (1996).
Cohen et al. 1990	P.R. Cohen, J. Morgan, and M.E. Pollack (eds), "Intentions in Communication", MIT Press, Cambridge, Massachusetts.
Cohen and Levesque 1990	P.R. Cohen and H.J. Levesque, "Rational Interaction as the Basis for Communication", in Cohen et al. (1990).
Cooper et al. 1999	R. Cooper, S. Larsson, C. Matheson, M. Poesio, and D. Traum, *"Coding in Structural Dialogue for Information States"*. Deliverable D1.1. Trindi Project, 1999.
Colby 1973	K. Colby, *"Simulations of Belief Systems"*, In Schank and Colby (1973).
Eckert and McGlashan 1993	W. Eckert and S. McGlashan, “*Managing Spoken Dialogues for Information Services”,* in Proceedings of Eurospeech, Berlin, Germany, 1993.
Elsendoorn and Bouma 1989	B.A. Elsendoorn and H. Bouma, *“Working models of Human Perception”*, Academic Press, 1989.
Fishman 2000	B. Fishman and S. O'Connor-Divelbiss (eds.), *Proceedings of the Fourth International Conference of the Learning Sciences*,. Mahwah, NJ: Erlbaum, 2000.
Grosz and Sidner 1986	B.J. Grosz and C.L. Sidner, *“Attentions, Intentions, and the structure of Dialogue”*, Computational Linguistics, 12(3): 175-204.
Huang et al. 1993	X.D. Huang, F. Alleva, H.W. Hon, M.Y. Hwang, K.F. Lee, and R. Rosenfeld, “*The SPHINX-II Speech Recognition System: An Overview”, Computer, Speech and Language*, 2, 1993. Also published as *Technical Report CMU-CS-92-112, School of Computer Science, Carnegie Mellon University*, Pittsburgh, PA, February, 1992.
Hulstijn et al. 1996	J. Huilstijn, R. Streetskamp, H. ter Doest, S. van de

	Burgt, and A. Nijholt, *"Topics in SCHISMA Dialogues"*, in Luperfoy et al. (1996).
Jefferson, 1972	G. Jefferson, *"Side Sequences"*, in Sudnow (1972).
Kowtko et al. 1991	J.C. Kowtko, S.D. Isard, and G.M. Doherty, *"Conversational Games within Dialogue"*, in Proceedings of the ESPRIT Workshop on Discourse Coherence, University of Edinburgh, 1991.
Lamel et al. 1999	L. Lamel, S. Rosset, J.L. Gauvain, and S. Bennacef, *"The LIMSI Arise system for Train Travel Information"*, in IEEE International Conference On Acoustics, Speech, and Signal Processing*,* Phoenix, 1999.
Lamel et al. 2000	L. Lamel, S. Rosset, J.L. Gauvain, S. Bennacef, M. Garnier-Rizet, and B. Prouts, *"The LIMSI ARISE System"*, in *Speech Communication*, 31(4):339-354, 2000.
Larsson 2000	S. Larsson and D. Traum, *"Information State and Dialogue Management in the TRINDI Dialogue Move Engine Toolkit"*, in NLE Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering, 2000.
Lee and Wilks 1996	M. Lee and Y. Wilks, *"An Ascription Based Approach to Speech Acts"*, in proceedings of the 16th Conference on Computational Linguistics (COLING-96), Copenhagen, Denmark, 1996.
Lemon 2001	O. Lemon, A. Bracy, A.r Gruenstein, and S. Peters, *"The Witas Multi-Modal Dialogue System I"*, in proceedings of Eurospeech2001, Aalborg (Denmark), 2001.
Levy et al.1997	D. Levy, R. Catizone, B. Battacharia, A.Krotov and Y. Wilks, *"CONVERSE: A Conversational Companion"*, in Proceedings of the 1st International Workshop on Human-Computer Conversation, Bellagio, Italy, 1997.
Luperfoy et al. 1996	S. Luperfoy, A. Nijholt, and G. Veldhuijzen van Zanten (eds.), *"Dialogue Management in Natural Language Systems"*, Proceedings of the 11th Twente Workshop on Language Technology, University of Twente, Enschede, Netherlands. 1996.
Matheson et al. 2000	C. Matheson, M. Poesio, and D. Traim, *"Modelling Grounding and Discourse Obligations Using Update Rules"*, in Proceedings of the First Conference of the North American Chapter of the Association for Computational Linguistics, 2000.
McDermott 1981	D. McDermott, "*Artificial Intelligence Meets Natural Stupidity*", in "Mind Design: Philosophy, Psychology, Artificial Intelligence", J. Haugeland (ed), MIT Press, 1981.
McGlashan et al. 1992	S. McGlashan, N.M. Fraser, G.N. Gilbert, E. Bilange, P. Heisterkamp, and N.J. Youd, "*Dialogue Management for Telephone Information Systems"*, in Proceedings of the

	International Conference on Applied Language Processing, Trento, Italy, 1992.
McGlashan 1996	S. McGlashan, *“Towards Multimodal Dialogue Management”*, in Luperfoy et al. (1996).
Moeschler 1989	J. Moeschler, *“Modélisation du dialogue, représentatione de l’inférence argumentative”*, Hermes.
Müller and Runger 1993	C. Müller and F. Runger, *"Dialogue Design Principles - Key for Usability of Voice Processing"*, in Proceedings of the 3rd European Conference on Speech, Communication, and Technology (EUROSPEECH93), Berlin, Germany, 1993.
Nielsen and Baekgaard, 1992	Nielsen and A. Baekgaard, "*Experience with Dialogue Description Formalism for Realistic Applications"*, in Proceedings of the International Conference on Spoken Language Processing (ICSLP 92), Banff, Canada, 1992.
Novick and Hansen 1995	D.G. Novick and B. Hansen, *"Mutuality Strategies for Reference in Task-Oriented Dialogue"*, in Andernach et al. (1995).
Novick and Ward 1993	D.G. Novick and K. Ward, *"Mutual Beliefs of Multiple Conversants: A Computational Model of Collaboration in Air Traffic Control"*, in Proceedings of AAAI'93, 1993.
Person and Graesser 2000	N.K. Person and A.C. Graesser, *“Designing AutoTutor to be an Effective Conversational Partner”*, in Fishman 2000.
Polany and Scha, 1984	R. Polany and R. Scha, *"A Syntactic Approach to Discourse Semantics"*, in Proceedings of the 10th International Conference on Computational Linguistics, Stanford University, California, ACL, 1984.
Reichman 1981	R. Reichman, *"Plain-Speaking: a Theory and Grammar of Spontaneous Discourse"*, PhD thesis, Department of Computer Science, Harvard University, Cambridge, Massachusetts, 1981.
Schank 1981	R.C. Schank and C.K. Riesbeck, "Inside Computer Understanding", Hillsdale, NJ: Lawrence Erlbaum
Schank and Colby 1973	R.C. Schank and K.M. Colby (eds.), *"Computer Models of Thought and Language*", Freeman, San Francisco, CA, 1973.
Searle 1969	J.R. Searle, *"Speech Acts: An Essay in the Philosophy of Language"*, Cambridge, University Press, 1969.
Sidner 1992	C.L. Sidner, *"Using Discourse to Negotiate in Collaborative Activity: an Artificial Language"*, in AAAI-92 Workshop *"Cooperation Among Heterogeneous Intelligent Systems"*, 1992.
Sidner 1994	C.L. Sidner, *"An Artificial Disocurse Language for Collaborative Negotiation"*, in proceedings of the AAAI,

	1994.
Sinclair and Coulthard 1975	J.M. Sinclair and M. Coulthard, *"Towards an analysis of disourse: the English used by teachers and pupils"*, Oxford University Press, 1975.
Sturm et al. 1999a	J. Sturm, E.A. den Os, and L. Boves, *"Issues in Spoken Dialogue Systems: Experiences from the Dutch Arise Train Timetable Information System"*, in Proceedings ESCA Workshop on Interactive Dialogue in Multi-Modal Systems, Kloster Irsee, Germany, 1999.
Sturm et al. 1999b	J. Sturm, E.A. den Os, and L. Boves, *"Dialogue Management in the Dutch Arise Train Timetable Information System"*, in Proceedings Eurospeech '99, Budapest, Hungary, 1999.
Sudnow 1972	D. Sudnow (ed.), *"Studies in Social Interaction"*, New York, The Free Press, 1972.
Traum 1996	D.R. Traum, *"Conversational Agency: The TRAINS-93 Dialogue Manager"*, in Luperfoy et al. 1996.
Veldhuijzen van Zanten 1996	G. Veldhuijzen van Zanten, *"Pragmatic Interpretation and Dialogue Management in Spoken-Dialogue Systems"*, in Luperfoy et al. (1996).
Wahlster et al. 2001	W. Wahlster, N. Reithinger, and A. Blocher, *"SmartKom: Multimodal Communication with a Life-Like Character"*, in Proceedings of Eurospeech2001, Aalborg (Denmark), 2001.
Weizenbaum 1966	J. Weizenbaum, *"ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine*", Communications of the Association for Computing Machinery 9, 1966.
Wilks and Ballim 1989	Y. Wilks and A. Ballim, "Shifting the belief engine into higher gear", in Proceedings of the International Conference on AI Methodology Systems Applications, (AIMSA '88), T. O'Shea and V. Sgurev (eds.). Amsterdam: Elsevier, 1989.
Wilks and Ballim 1991a	Y. Wilks, and A. Ballim, *"Artificial Believers"*, Norwood, NJ: Erlbaum. 1991.
Wilks and Ballim 1991b	Y. Wilks, and A. Ballim, *"Beliefs, Stereotypes and Dynamic Agent Modeling"*, In *"User Modeling and User-Adapted Interaction"*, Vol.1, No. 1, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991.
Wilks 1999	Y. Wilks, *"Conversations with a belief system"*. In Proceedings of the Conference on Natural Language Processing and Medical Concept Representation, IMIA Working Group 6. Phoenix, AZ, 1999.
Wilks and Catizone 2001	R. Wilks, and R. Catizone, *"Human-Computer Conversation"*, Encyclopedia of Library and Information Science, Vol. 69, Allan Kent (ed.). New York: Dekker, 2001.

Williams 1996	S. Williams, *"Dialogue Management in a Mixed-Initiative, Cooperative, Spoken Language System"*, in Luperfoy et al. (1996).
Woods 1979	W.A. Woods, "Transition Network Grammars for Natural Language Analysis", in CACM 3(10), 1970.o
Zinn 2002	C. Zinn, J. D. Moore, and M. G. Core, **"***A 3-tier Planning Architecture for Managing Tutorial Dialogue"*, in Intelligent Tutoring Systems, 6th. International Conference, ITS 2002, Springer LNCS 2363, Biarritz, France, June 2002.